

# **Network modularity and local environment similarity as descriptors of protein structure**

William Peter Grant

Theory of Condensed Matter Group  
Cavendish Laboratory



This dissertation is submitted for the degree of  
Doctor of Philosophy

December 2019, Trinity College  
Supervised by Dr. Sebastian E. Ahnert

## Declaration

This dissertation is the result of my own work and conforms to the University of Cambridge's guidelines on plagiarism. Where reference has been made to other research this is acknowledged in the text and bibliography. This dissertation does not exceed the word limit for the respective Degree Committee.

Sections from Chapters 4 and 5 of the thesis are based on research published in:

William P Grant, Sebastian E Ahnert, "Modular decomposition of protein structure using community detection", Journal of Complex Networks, Volume 7, Issue 1, February 2019, Pages 101-113.  
[doi.org/10.1093/comnet/cny014](https://doi.org/10.1093/comnet/cny014)

Other work published during my doctoral research not included in this thesis:

Sebastian E Ahnert, William P. Grant, Chris J. Pickard, "Revealing and Exploiting Hierarchical Material Structure through Complex Atomic Networks", npj Computational Materials 3.1 (2017)  
[doi.org/10.1038/s41524-017-0035-x](https://doi.org/10.1038/s41524-017-0035-x)

The code described in this work can be found at:

<https://github.com/ProteinNetworks/>

William Grant



## Acknowledgements

I owe this thesis to the support and kindness of a huge number of people, not all of whom I can thank here. My parents have supported and encouraged me unconditionally throughout my education, from the first day of primary school aged four until the submission of my PhD aged twenty-seven. I'd like to thank my fellow members of TCM IT Support, Mark Johnson and Matthew Evans, for keeping me sane throughout the PhD process. I'd also like to thank Matthew for his help in proof-reading, as well as printing, binding, and submitting the thesis on my behalf. Hannah Meyer provided valuable comments on the thesis draft. I'd like to thank Michael Rutter for teaching me how computers work.

I'd like to thank my supervisor, Sebastian, for his advice and support.

Finally, I'd like to thank my wife, Christina, for her unfailing kindness, her superb insight, and for reminding me what the whole endeavour is for.

## Abstract

As the number of solved protein structures increases, the opportunities for meta-analysis of this dataset increase too. Here we explore two approaches for analysing protein structure, both starting from the three-dimensional co-ordinates of each atom within the structure, which are then abstracted into a more useful form.

The first method transforms the protein into a network in which its amino acids are the nodes, and where the edges are generated using a simple proximity test. By applying the Infomap community detection algorithm, we can fragment the protein into highly intra-connected subregions - these subregions are compact and globular, and can be compared with known structural and functional subunits of the protein (also known as domains). By performing this fragmentation process systematically across a large set of proteins, and checking for structurally conserved fragments, we can search for novel candidate domains. This method for automatically decomposing a protein into compact substructures may also be useful in coarse-graining molecular dynamics, analysing the protein's topology, in de novo protein design, or in fitting electron density maps derived from single particle electron microscopy.

The second method calculates a descriptor for each atom of the protein based on its local environment, known as a Smooth Overlap of Atomic Positions (SOAP) descriptor. Using these descriptors we can perform overall comparisons of the subregions identified above. In addition, by comparing the descriptors of a set of proteins known to share common structural or functional features (such as binding of a particular ligand), we can automatically identify the most highly conserved atoms of the set. These atoms may line ligand binding pockets or correspond to allosteric sites, which could inform drug design.

# Contents

<b>1</b>	<b>Motivation</b>	<b>1</b>
<b>2</b>	<b>Background theory</b>	<b>3</b>
2.1	Proteins . . . . .	3
2.2	Smooth Overlap of Atomic Positions . . . . .	9
2.3	Networks . . . . .	13
2.4	Community structure . . . . .	18
2.5	Previous work on protein structure networks . . . . .	27
<b>3</b>	<b>Codebase development</b>	<b>29</b>
3.1	Software architecture . . . . .	29
3.2	Network generation . . . . .	30
3.3	Community detection . . . . .	42
<b>4</b>	<b>Communities as conserved modules</b>	<b>47</b>
4.1	Defining the modified Jaccard index . . . . .	47
4.2	Collating existing annotation schemes . . . . .	50
4.3	Comparing communities to Pfam annotation . . . . .	50
4.4	Comparing communities to SCOP annotation . . . . .	54
4.5	Communities as novel candidate domains . . . . .	60
4.6	Comparison to existing methods . . . . .	67
4.7	Community structure at a sub-domain length scale . . . . .	68
<b>5</b>	<b>Community structure as a proxy for topology</b>	<b>73</b>
5.1	Generating and describing protein super-networks . . . . .	73
5.2	The properties of protein super-networks . . . . .	74
5.3	Network vs structure similarity . . . . .	77
5.4	Iterative subtree similarity . . . . .	80
<b>6</b>	<b>Application of SOAP to binding sites</b>	<b>85</b>
6.1	Classifying proteins according to secondary structure composition . . . . .	86
6.2	Application to Binding Pockets . . . . .	89
<b>7</b>	<b>Discussion</b>	<b>99</b>

---

<b>Appendix</b>	<b>101</b>
A.1 Scaling of the number of partitions with network size . . . . .	101
A.2 The AFG algorithm . . . . .	102
A.3 Network properties of test proteins . . . . .	105
A.4 Example communities for the filtered ASTRAL dataset . . . . .	106
<b>Bibliography</b>	<b>112</b>
<b>List of Figures</b>	<b>122</b>

# Chapter 1

## Motivation

Proteins are key to all cellular life, as they form the structures responsible for almost all functions within the cell. These structures consist of long chains of covalently bonded amino acids. These chains then fold into a shape which minimises their free energy, either freely or with the help of chaperone proteins. Understanding the structure of a given protein can give insight into its function within the cell, and suggest drug targets for modifying this function. Understanding the organisational and evolutionary principles behind protein structure as a whole may be more valuable still.

Determining a protein's structure is done experimentally, using X-ray crystallography, Nuclear Magnetic Resonance, or cryo-electron microscopy (EM). The majority of structures have historically been solved using X-ray diffraction, though recent advances in EM may result in it becoming the dominant method. In both cases, it is a costly process in terms of time and human effort, and cannot be done systematically, as the exact conditions required to crystallise a protein must be determined empirically. However, over 130,000 proteins now have publicly available structures, and the size of this dataset is growing exponentially [1]. We need automated tools to be able to find common patterns or principles in a dataset of this scale.

There have been many past efforts at protein structure meta-analysis, using topological similarity [2] or evolutionary homology [3], in order to understand common origins and help infer function from structure. The most widely used structural databases search for repeated sub-structures in sets of proteins, and attempt to tie this to a given cellular function.

This work aims to assess the potential for describing protein structure globally in terms of spatial networks of amino acids, and locally in terms of each atom's specific environment.

Both descriptions are based on an abstracted form of the 3D co-ordinates of the individual atoms.

In the first, we abstract the protein into a network. We let the atoms or amino acids of the protein be vertices in the network, and apply a distance threshold to generate edges. Converting the protein structure into a network allows us to use pre-existing network analysis methods to examine the pattern of bonding within the protein.

The specific network property investigated in this work is community structure [4–7]; the idea that many networks are formed of highly intra-connected sub-networks, weakly connected to each other. These communities have been shown to be functionally relevant in many different systems, from finance networks to food webs [8–11].

Whilst network-based descriptions have shown promise in previous work in predicting key amino acids within proteins for allostery, and for protein stability [12–14], no previous work has systematically analysed the community structures of large sets of proteins, either as a method for extracting candidate protein domains (Chapter 4) or as a tool for investigating protein topology (Chapter 5).

In the second, we use the Smooth Overlap of Atomic Positions (SOAP) descriptor. This method describes an atom not by its 3D location in space, but by describing the relative position and identity of its neighbours, in a rotationally-invariant way. These descriptors have been successfully used in condensed-matter physics in the fitting of potentials, and in classification problems on ligands and crystal structures [15–17]. Here we extend this method to sets of protein structures, with the specific goal of discovering binding pockets (Chapter 6).

# Chapter 2

## Background theory

In this chapter the biology and bioinformatics required for this work are introduced, including a summary of existing protein classification methods. This is followed by an overview of the relevant areas of network science. Pre-existing work on the application of network analysis to protein structure will also be discussed. Finally, the theory underlying the Smooth Overlap of Atomic Positions (SOAP) descriptor will be given.

### 2.1 Proteins

In each human cell there are roughly 2 billion proteins [18]. Virtually every function or task performed by a cell is carried out by a protein, from enzymes which catalyse biological reactions, to membrane proteins allowing transport of material into and out of the cell. The diverse range of possible cellular functions is reflected in the dizzying variety of protein shapes; each protein fulfils a specific role, and this role is determined by its structure.

All proteins are formed from a chain of amino acids, covalently bonded. There are only twenty types of amino acids found within most eukaryotes, and yet the amino acid sequence is enough to uniquely specify the protein's structure [19]. Each amino acid has different properties, such as charge or hydrophobicity, and the interaction between these amino acids causes the chain to fold into a specific shape.

A protein's sequence is specified within the cell's DNA. Each amino acid is specified by a set of three nucleotides, known as a codon. The sequence of DNA base pairs is transcribed into mRNA, and then this mRNA sequence is translated into a sequence of amino acids by the ribosome. This amino acid chain then folds into the required shape, often with the help of chaperone proteins [20]. The direction of the flow of genetic information, from

DNA to RNA and from RNA to protein, is known as The Central Dogma [21]. In the human genome around 30,000 proteins are encoded (ignoring alternative splicing, which results in alternative processing of one mRNA such that a number of related proteins are produced), though not all are expressed in each cell [22].

Whilst the underlying chemistry is well understood, the way in which this gives rise to the final structure is not. Predicting the structure of a protein from its sequence has long been an active area of research. However, with over 130,000 protein structures now known, the problem of inferring function from structure has acquired new importance. In this section the relevant features of protein biochemistry are summarised, followed by an overview of the attempts within bioinformatics to perform meta-analysis of protein structure. Finally, we detail existing applications of network science to proteins.

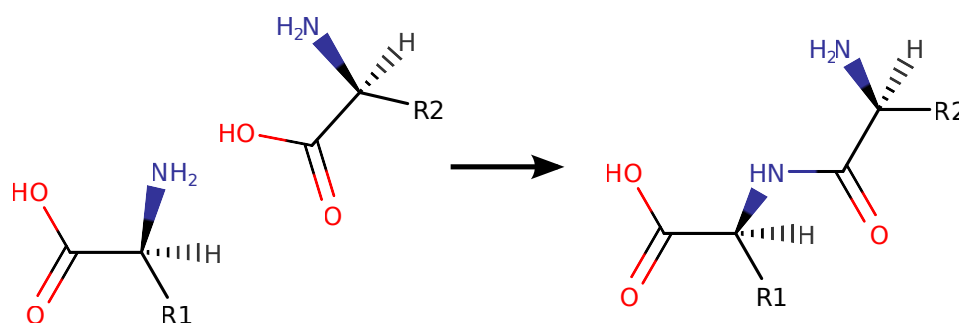
## Protein biochemistry

Despite the extreme variety in structures and functions of proteins, all proteins share a common synthesis pathway. In addition, almost all eukaryotes have a repository of only twenty possible amino acid building blocks. This means that the proteins obey common chemistry, and can be compared at several length scales, as shown below.

The fundamental building block of the protein is the amino acid. This is an organic compound comprised of a central carbon atom, labelled  $C_\alpha$ , bearing four functional groups; an amino group ( $-\text{NH}_2$ ), a carboxyl group ( $-\text{COOH}$ ), a hydrogen, and a variable group known as the side-chain (see Figure 2.1). At physiological pH, both of the amino- and carboxyl- groups are ionised. The amino group of one amino acid can bond covalently with the carboxyl group of another, creating a strong C-N bond known as a *peptide bond*. This process can repeat, generating a polypeptide chain. This chain is directional, with an unbound amino group at one end (the N-terminus) and an unbound carboxyl group at the other (the C-terminus). The bonded amino acids are known as *residues*.

The twenty possible amino acids are distinguished by their side-chain. The amino acid's side-chains can have different chemical properties (acidic, basic, polar, nonpolar, hydrophobic, hydrophilic), which enable them to interact with each other non-covalently and independently of their position in the polypeptide chain. These non-covalent interactions give rise to a protein's unique structure. Changing the amino acid sequence changes the chemical properties of the side-chain at a given position. This in turn changes the non-covalent bonding pattern across the residues, and hence its structure. It has





**Figure 2.1:** Schematic representation of amino acids, showing bond formation. R1 and R2 are the side-chains which vary between amino acids, and can be one of 20 possible structures. Figure taken from [23].

been shown that the ground state of the protein (its native conformation) is uniquely determined by its sequence [19].

There are four types of non-covalent interaction determining the folding of the covalently bonded chain:

- **Hydrogen Bonding:** Covalent N-H and O-H bonds are highly polarised, and as such a highly directional bond can form between the electropositive H on a donor atom and an electronegative acceptor atom. All amino acid residues contain possible hydrogen bond donors and acceptors making up the peptide bond, making repeated hydrogen bonds between sections of the polypeptide chain feasible. Hydrogen bonding between water and the side-chains is also possible.
- **Ionic Bonding:** Several of the possible side-chains are ionised in physiological conditions, either negatively charged (aspartic acid, glutamic acid) or positively charged (arginine, histidine, lysine). These charges are screened by water, but electrostatic interaction between such charges still occurs.
- **Van-der-Waals Bonding:** Electron density will fluctuate with time; this can result in spontaneous dipole interactions between non-polar atoms. This interaction is weak, but significant when two surfaces come into close contact.
- **Hydrophobicity (i.e Entropy):** Bulk water will form a constantly shifting network of hydrogen bonds, with high entropy. A large nonpolar molecule will disrupt this network, as a solvation shell around the molecule is formed. This shell reduces the mobility of the water molecules, and hence the entropy of the system. This entropic cost causes non-polar side-chains to be preferentially buried within the interior of the protein.

These interactions are 10-100 times weaker than the typical covalent bond [24], but are much more numerous. In addition, proteins are metastable. The energies of possible conformations often differ by only the energy of a few hydrogen bonds.

## Levels of Protein Structure

Biologists classify the structures formed as a result of the above interactions into four discrete levels:

- **Primary Structure:** The structure at the amino-acid level, i.e. the linear sequence of the residues within the polypeptide chain.
- **Secondary Structure:** The N-H and C=O groups in each amino acid can form hydrogen bonds. By considering the steric constraints on the peptide bond, and the possible interactions between different elements of the chain, it can be shown that there are two (and only two) highly regular patterns of hydrogen bonding [25,26]. These are known as  $\alpha$ -helices, and  $\beta$ -sheets. These structures require no solvent, and no specific side-chain, so often form within the core of the protein.  $\alpha$ -helices are formed when a single polypeptide chain coils to form a rigid cylinder, such that every fourth peptide bond forms a hydrogen bond in a right-hand helix. A  $\beta$ -sheet forms when two sections of chain run either parallel or anti-parallel to each other. The arrangement of these regular substructures defines the secondary structure.
- **Tertiary Structure:** The full three-dimensional arrangement of a single, folded, polypeptide chain.
- **Quaternary Structure:** Often a protein is formed from several polypeptide chains as a non-covalently bonded complex. The canonical example is haemoglobin, which forms a single functional unit made of four chains [27]. The quaternary structure is the relative positioning of these component polypeptide chains.

In addition to these levels, the protein *domain* is a region of sequence or structure repeated across groups of proteins, normally 40-350 residues in size [24]. Protein structure domains are compact, globular subregions within the protein that are normally associated with a specific function - likewise, protein sequence domains tend to be contiguous regions of the polypeptide chain, though sequence and structure domains may not overlap. Protein evolution is often modelled as the threading of beads (domains) on a string (the polypeptide chain), with a protein acquiring additional functionality by incorporating the relevant domain.

## Protein classification

There are roughly 125,000,000 unique protein sequences stored in UniProtKB [28], and roughly 130,000 protein structures stored in the Protein Data Bank [29]. The disparity in the size of the datasets is due to disparity in the effort required; structure determination is carried out either using nuclear magnetic resonance (NMR), electron cryo-microscopy, or X-ray diffraction. This is extremely costly in time and effort, and cannot be done systematically. In contrast, a protein's sequence can be determined using DNA sequencing of the relevant organism. Advances in sequencing mean that a genome can typically be mapped for under £1000, and in under 1 day [30,31]. If the gene encoding a given protein is known, then the amino acid sequence can be inferred.

The size of these datasets is growing at an exponential rate [1]. Whilst individual structures may give valuable insight into specific biochemical pathways, and present crucial drug targets, the principles behind protein structure evolution cannot be understood on a case-by-case basis. Many attempts have been made to group the set of solved protein sequences and structures, in order to understand evolutionary relationships or elucidate function. Here a selection of such databases are given. All are based on the idea of domains as defined previously; regions of sequence or structure repeated across groups of proteins, normally 40-350 residues [24]. These domains have traditionally been found using manual curation. Automated methods often use Hidden Markov Models (HMMs) [32] which find repeats across the full range of structures, yet cannot infer whether these repeats have any functional meaning.

**Pfam:** The Pfam database [33] is a collection of protein families, which classifies proteins by sequence. Sequence domains are found using HMMs, giving conserved regions across sets of proteins. This gives insight into protein evolution. There are currently over 16,000 PFAM families, covering >80% of the total sequence set stored in UniProtKB.

**SCOP:** The Structural Classification of Proteins (SCOP) database [2,34] is a hierarchical protein structure classification scheme, grouping proteins by shared regions of structure. The set of known protein structures is separated by Class; proteins within a class have large-scale structural differences, but have a similar composition of secondary structure. For example, class a of the SCOP database corresponds to fully  $\alpha$ -helical proteins. Each class is split into Folds; proteins within a Fold share regions of structural similarity, though without clear evidence of evolutionary similarity. Each Fold is split into a Superfamily; within a Superfamily structural and functional attributes suggest a common evolutionary

origin, though without sequence similarity. The lowest level of hierarchy is the Family; proteins within a family will have significant sequence and structural similarity.

**CATH:** The Class, Architecture, Topology, Homology database [35] is, similarly to SCOP, a hierarchical structural classification. Three independent domain identification methods (PUU [36], DETECTIVE [37] and DOMAK [38]) identify structural domains. The classification is more geometric and less functional than SCOP.

**ECOD:** The Evolutionary Classification of Domains database [3] is also a structural database, but differs from SCOP and CATH in that its groupings are evolutionary rather than topological, with an emphasis on distant evolutionary relationships that are otherwise difficult to detect.

Another key resource is the Gene Ontology (GO) project [39]. This uses a directed acyclic graph to assign attributes (e.g. lactase activity, oxidoreduction) to genes. This has traditionally been used in gene sequencing efforts, in which the enrichment (i.e. the occurrence at a greater rate than predicted by chance) of given terms may give insight into the function of a given protein variant [40,41].

However, given that genes encode proteins, this implies that for each protein in the PDB there is a set of associated GO terms. This mapping has been carried out by SIFTS [42], and can be used to assign functions to proteins.

## Protein similarity algorithms

To find patterns in a large set of proteins, such as conserved domains or binding sites, we need a metric for the similarity of two proteins. More fundamentally, in order to generate such a metric, we need a way of describing the protein in a machine-parsable way. There are two widely-used classes of descriptors:

- Those based on the proteins' sequences. Here the descriptor is a string of characters representing the amino acid sequence.
- Those based on the proteins' structure. Here the descriptors are the spatial positions of each atom in the protein, as a vector of 3D co-ordinates.

Here we focus on descriptions based on the proteins' structure.

The earliest methods for comparing two sets of co-ordinates are based on alignment [43]. By arranging the proteins so as to minimise the distance between pairs of equivalent

atoms, we can obtain the root-mean-square distance (RMSD) between the two proteins' atoms. This RMSD then functions as a similarity score.

The problem of protein alignment is NP-hard [44], and is especially challenging when the proteins have different lengths or have highly differing sequences (as the notion of 'equivalent pairs' of atoms then becomes difficult to define). The RMSD measure also has the effect of prioritising local over global similarity, with more recent scoring methods such as the TM-score [45] using an adjusted metric to compensate for this.

A more recent class of scoring methods forgoes alignment and the absolute positions of the atoms, and instead compares the positions of the atoms in the protein relative to each other. By creating an  $N$  by  $N$  matrix of the distance from each atom in the protein to every other atom in the same protein, we generate a contact map. By using matrix comparison methods we can thus compare the structures of two proteins [46–48].

For a more coarse-grained representation, the spatial positions of the amino acids, as encoded by the  $C_\alpha$ , can be used instead.

In this work, we apply two classes of protein structure descriptors; one based upon local environment similarity, and one based on spatial networks. In section Section 2.2, the mathematics behind the SOAP descriptor is introduced, and in Section 2.3 the relevant network science.

## 2.2 Smooth Overlap of Atomic Positions

In Chapter 6 we apply a descriptor based on the local environment of each atom within the structure, i.e. the overall arrangement of the atom's neighbours. This approach is distinct from both the comparison of absolute atomic positions (as in alignment), or the relative position between atoms (as in the spatial network approach). These local environment descriptors have shown promise in solid-state physics when used to train models on inorganic crystal structures or protein ligands [16, 17], but have yet to be applied to protein structure itself. This descriptor is known as the Smooth Overlap of Atomic Positions (SOAP) descriptor. The derivation below summarises work given in [17].

For each atom, we get the position of every other atom within a certain cutoff distance  $R$  - these atoms are defined as belonging to environment  $\chi$ .

We then represent each atom  $i$  within this environment with a Gaussian with variance  $\sigma^2$ , centred on  $x_i$ , the atom's original position.

The local density in environment  $\chi$  is then:

$$\rho_\chi(r) = \sum_{i \in \chi} \exp\left(-\frac{(x_i - r)^2}{2\sigma^2}\right)$$

The SOAP similarity kernel between two atoms with local environments  $\chi$  and  $\chi'$  is then defined as the overlap between the two densities  $\rho_\chi(r)$  and  $\rho_{\chi'}(r)$ , integrated over all three-dimensional rotations  $\hat{R}$ .

$$k(\chi, \chi') = \int d\hat{R} \left| \int \rho_\chi(r) \rho_{\chi'}(\hat{R}r) dr \right|^2$$

It can be shown [15] that this integral can be performed analytically. If the local environment density is expanded in a basis of spherical harmonics  $Y_{lm}$  and radial functions  $g_b$ :

$$\rho_\chi(r) = \sum_{blm} c_{blm} g_b(|r|) Y_{lm}(\hat{r})$$

Then the power spectrum is defined as

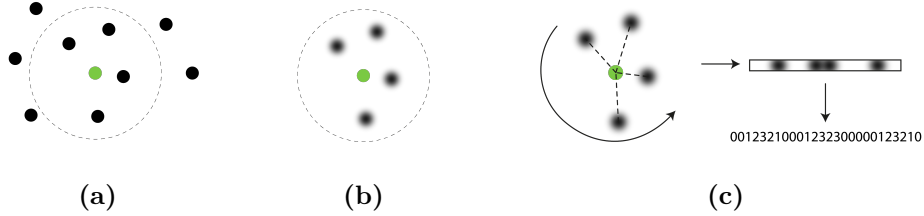
$$p(\chi)_{b_1 b_2 l} = \sum_m (c_{b_1 l m})^\dagger c_{b_2 l m}$$

If these elements of the power spectrum are collected as a unit-length vector  $\hat{p}$ , the SOAP similarity kernel between atomic environments  $\chi$  and  $\chi'$  can be shown to be [15]:

$$k(\chi, \chi') = \hat{p}(\chi) \cdot \hat{p}(\chi')$$

The practical result of this surprising mathematical equivalence between the power spectrum and the rotational overlap integral is that the SOAP descriptor for an atom

can be considered to be this unit-length vector  $\hat{p}$ , with the similarity score between two atoms simply equal to the inner product between their SOAP descriptors.



**Figure 2.2:** Schematic of the SOAP descriptor generation process. (a) For the target atom in green, the atoms within a given cutoff radius are selected (as indicated by the dotted line). (b) For each atom within the cutoff radius, a Gaussian with variance  $\sigma^2$  is placed at the atom's position. (c) This set of Gaussians is expanded in terms of spherical and radius basis functions, and the power spectrum calculated. The coefficients of the power spectrum, as a vector, give us the SOAP descriptor.

Once the SOAP descriptors for each of a protein's atoms have been calculated, they can then be used to calculate global similarity between two proteins, as follows.

## Global Similarity (GLOSIM)

Given the set of SOAP descriptors for two structures  $A$  and  $B$ , we can calculate the overall (or global) similarity between  $A$  and  $B$  in one of three ways [16]. The GLOSIM measure of overall similarity between two structures will be used in Chapter 4 to compare protein fragments.

For each atom  $x_i$  in structure  $A$ , and atom  $x_j$  in structure  $B$ , we define the SOAP kernel as  $k(x_i, x_j)$ .

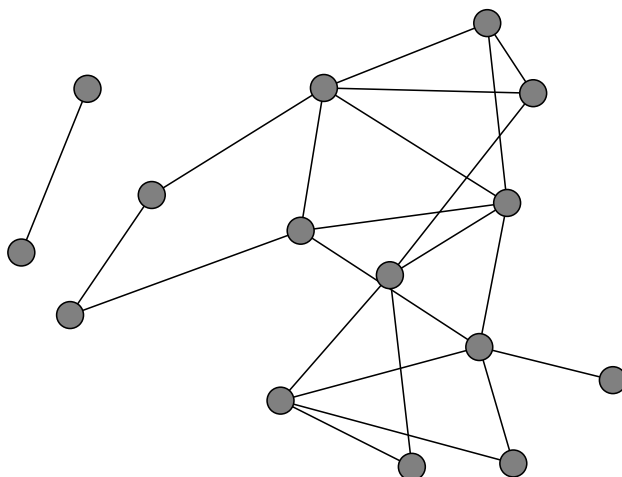
1. **Average:** The simplest method is to average over all possible pairs of environments:  $K(A, B) = \sum_{i \in A, j \in B} P_{ij} k(x_i, x_j)$  where  $P$  is the permutation matrix mapping  $i$  to  $j$ .
2. **MATCH:** Alternatively, we can find the permutation matrix  $P$  such that each  $i$ - $j$  pair has the highest  $k(x_i, x_j)$ . This can be computed in polynomial time using the Hungarian algorithm [49].
3. **Regularized MATCH (REMATCH):** Using a regularized entropy, we can smoothly interpolate between the average and MATCH method above (see [16] for details). Unlike the MATCH method, this can be applied in cases when  $A$  and  $B$  have different numbers of atoms.

---

In practice, we often wish to compare structures with different numbers of atoms, and REMATCH is fast enough that there is no need to use the less-accurate Average method, so the REMATCH kernel is used throughout the rest of the work.



## 2.3 Networks



**Figure 2.3:** An example of a network with 15 nodes and 23 edges.

In Chapter 4 and Chapter 5 we investigate application of network science methods - specifically, community detection - to networks based on protein structure. In the remainder of this chapter the required theory is introduced. This closely follows previous MPhil work [23].

A network (also known as a graph) is a set of nodes (or vertices,  $V$ ), connected by links (or edges,  $E$ ), as in Figure 2.3.

The study of the intrinsic properties of networks is known as graph theory. Within the framework of graph theory, a graph is defined as a pair  $(V, E)$  where  $V$  is a set and  $E$  is a subset of  $V^{(2)} := \{\{x, y\} : x, y \in V, x \neq y\}$ , which is the set of unordered pairs of  $V$ .

The behaviour of these abstract objects has traditionally been a problem of pure mathematics, and over hundreds of years the investigation of these properties have led to some of the most elegant and incisive proofs in mathematics. However, it is the application of graphs to real-world problems that has seen graphs gain new relevance in recent years.

The nodes and links of a graph can be used to represent any system of interacting components; the nodes represent the components themselves, and the links some aspect of their interaction. Abstracting away the details of the system allow the patterns of interaction to be explored. This can help identify key components, key interactions, or key properties of the system as a whole. The application of graphs to understanding real-world systems is normally referred to as network analysis, and fields such as finance,

genetics, agriculture and sociology [50–52] have all seen network analysis yield new insights.

The field of network analysis has gained new prominence due to increasing availability of massive data sets, and increasing computational power. Analysis of these data sets has shown that many real-world networks share key properties, regardless of the underlying system.

## Network Representation

In order for different networks to be compared and analysed, a common format for their representation is required. One standard form for representing a network is an *edge list*. The nodes are labelled  $[1, N]$  in arbitrary order. Then by specifying the number of vertices,  $N$ , along with a list of the start and end points for each edge e.g.  $[(1,2), (1,3), (2,4), \dots]$ , we can completely specify the network. If there are no isolated nodes without edges, then the list of start and end points is sufficient. This form is convenient for reading and writing networks.

An alternative representation of a network takes the form of an *adjacency matrix*. This matrix  $\mathbf{A}$  is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ share an edge.} \\ 0 & \text{otherwise.} \end{cases}$$

The representation allows matrix methods to be applied to network analysis. For example, it has been shown that the eigenvalues of this matrix are linked to the network's connectivity [53].

Networks in which the edges have uniform importance, and in which an edge  $i \rightarrow j$  implies an edge  $j \rightarrow i$ , are known as *unweighted* and *undirected* networks respectively. In many cases the interactions of a system have varying strength; this data can be incorporated by giving the edges a *weight*. The adjacency matrix representation can be extended to weighted networks by letting  $A_{ij}$  be the weight of the edge from  $i$  to  $j$ . The edge-list representation can be extended by added a third term to each row of the list, representing the weight e.g.  $[(1, 2, 0.5), (1, 3, 2.0), (2, 4, 1.0), \dots]$ .

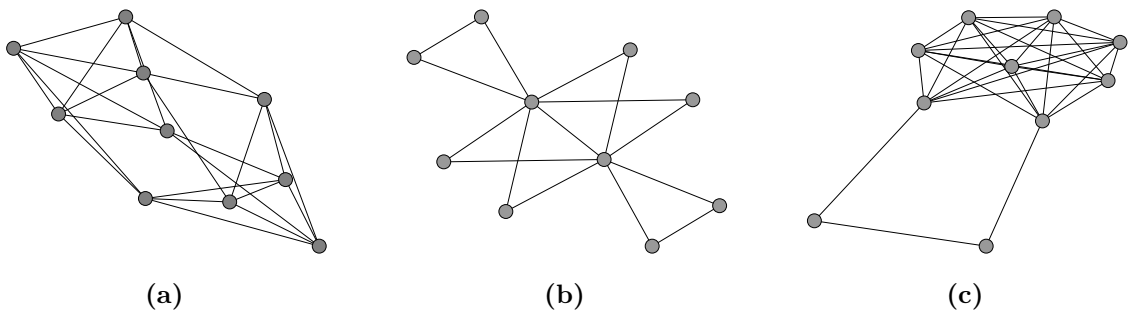
Additionally, there are many examples of systems in which the interactions are unidirectional, for instance in a network of citations [54], or hyperlinks within the Internet [55]. Networks for which this is true are known as *directed* networks. Note that for undirected networks the adjacency matrix is symmetric; for directed networks this is not generally true.

## Network Properties

Once a system has been converted to a network, the properties of interest are often dependent on the exact system, or the problem under study. However, in many systems what is required is some metric of node or edge importance. The simplest possible proxy for a node's importance is its *degree*; the number of edges connecting to it. In terms of the adjacency matrix, the degree of node  $i$ ,  $k_i$ , is:

$$k_i = \sum_{j=1}^n A_{ij}$$

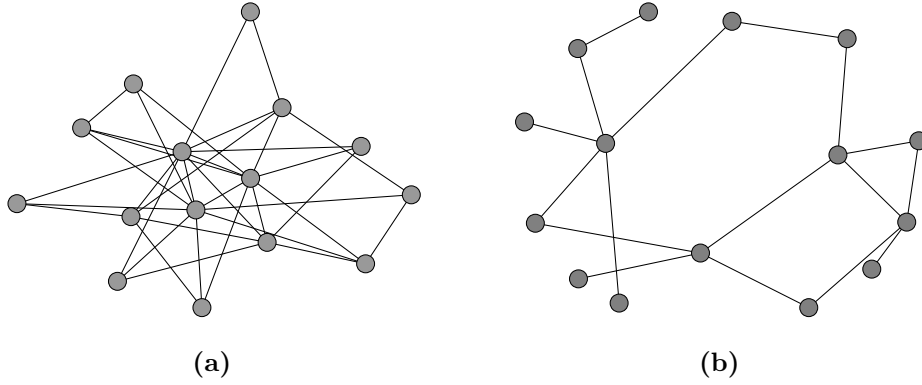
Nodes with high degree may occupy key positions within the network. The *degree distribution* is characteristic of the network.



**Figure 2.4:** Example networks showing three different degree distributions:  
 a) 10 nodes of degree 5 (i.e a uniform degree distribution).  
 b) 8 nodes of degree 2, and 2 nodes of degree 7 (i.e a low-skewed degree distribution).  
 c) 8 nodes of degree 7, and 2 nodes of degree 2 (i.e a high-skewed degree distribution).

Another network descriptor commonly of interest is the *average path length*. This defines how many edges separate each node from every other node in the system. Defining the *geodesic distance*,  $d_{ij}$ , as the shortest path from node  $i$  to node  $j$  (and  $:= 0$  if no such path exists). Then the average path length for the graph  $G$  is:

$$l_G := \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij}$$



**Figure 2.5:** Example networks showing low and high average path length. It can be seen that the path length corresponds to intuitive notions of connectivity.

a) a Barabási-Albert graph [56] with  $n=15$ ,  $m=3$ ,  $l_G=1.7$ .

b) an Erdős-Rényi graph [57] with  $n=15$ ,  $p=0.2$ .  $l_G$  is 2.9.

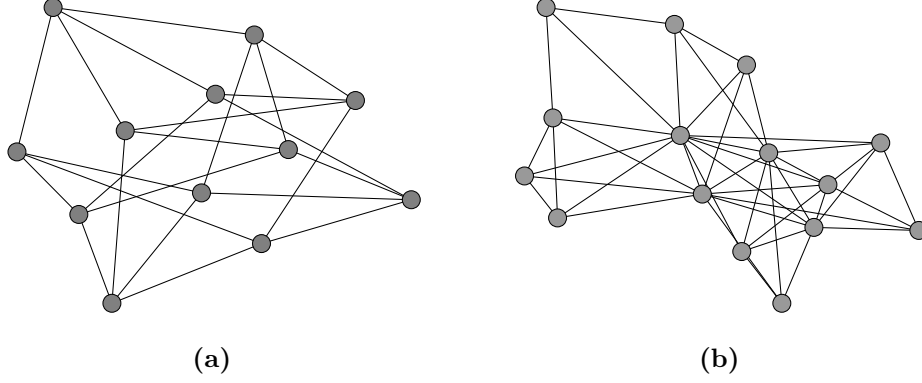
A third example of a network descriptor is the *clustering coefficient*, which details the likelihood that two connected nodes share a neighbour. The local clustering coefficient for a vertex  $i$  can be defined as the proportion of pairs of neighbours of  $i$  that are connected.

$$C_i := \frac{\sum_{j,k} A_{ij} A_{jk} A_{ki}}{\sum_{j,k} A_{ij} A_{jk}} = \frac{\sum_{j,k} A_{ij} A_{jk} A_{ki}}{\frac{1}{2} k_i (k_i - 1)}$$

We can then define a global clustering coefficient as an average over nodes:

$$C = \frac{1}{N} \sum_i C_i$$

These three properties are often characteristic of the type of network being studied. In addition, many networks show the same key form of given network parameters. For instance, many networks show a power-law distribution in the degree distribution, regardless of the underlying system,  $P(k) \sim k^{-\gamma}$ . Different types of networks will then exhibit different values of  $\gamma$  [50, 51, 59]. In addition, the average path length and the



**Figure 2.6:** Example networks showing low and high average clustering. Example (a) shows that the clustering coefficient can give unintuitive results, due to its dependence on triangles.

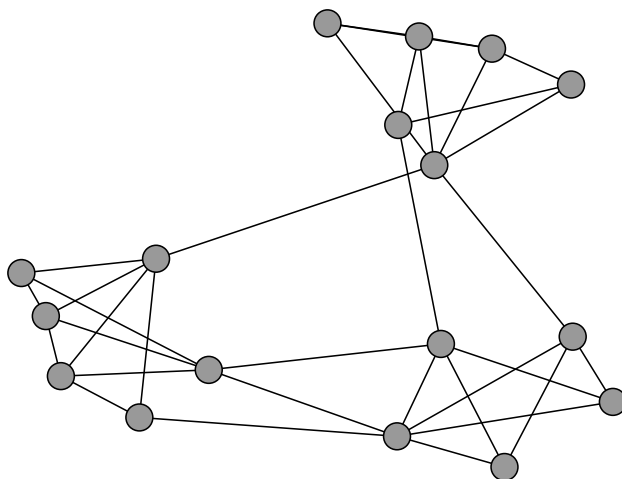
- a) The Chvátal graph [58], with clustering coefficient of 0.
- b) A graph with clustering coefficient 0.72.

clustering coefficient both deviate significantly from the values that would be expected by chance (the “Small-World Effect” [60]).

In addition to evaluating the properties of a network, we often wish to compare two networks to each other. Two networks are *isomorphic* if their nodes and edges match perfectly; that is, there is a relabelling of nodes of the first graph such that the edges of the second graph are recovered. Formally, networks  $G = (V, E)$  and  $H = (V', E')$  are isomorphic if there is a bijection  $f : V \rightarrow V'$  such that  $xy \in E \iff f(x)f(y) \in E'$ . It is not yet known whether graph isomorphism can be checked in polynomial time, or whether it is NP-complete; an algorithm has recently been proposed that may run in quasi-polynomial time [61].

A broader comparison can be given as a similarity measure between networks, also known as “fuzzy isomorphism”. This may also be recast in terms of the *edit distance*, the minimum number of edges that must be added or removed in order to convert one network to another. Alternative methods have focussed on graph fingerprinting; by creating a vector of characteristic graph properties we can compare graphs in terms of the Euclidean distance between their vectors. In Chapter 5 a novel subgraph isomorphism algorithm is developed, based on the matching subtrees.

## 2.4 Community structure



**Figure 2.7:** A network showing three communities, generated using the “relaxed caveman” model [62].

Many networks share a feature which cannot be extracted from either the local properties of each node, nor from the global properties of the network. This is the tendency of many observed networks to form *communities*; tightly bound collections of nodes, with few links to nodes outside the community (see Figure 2.7). Often, these communities correspond to relevant organisational units of the system. These communities have obvious meaning in the field of social science (corresponding to the traditional usage of the word “community”), but community structure has also been shown to be important in the the study of transport, biology, finance, and ecology, among others [9–11, 63, 64]. This community structure can be hierarchical, with multiple levels of nested communities [65–68].

Historically the realm of sociology [69–71], study of community detection by network scientists accelerated in 2002 with the introduction of modularity [4] as a metric for community strength. The field of community detection is today one of the busiest research areas in network science. Many comprehensive reviews of the field exist [5–7, 72–77]; here the salient features are presented, following previous MPhil work [23].

Community detection refers to the problem of extracting the community membership of a network, using only the nodes and edges of the graph (i.e. without any prior knowledge). This results in a *partition*, which is a list of each node’s community.

Two properties of communities are: They should be connected (i.e. there should be some path from every point in the community to every other point in the community), and

they should be locally dense (i.e. nodes in a community should be more likely to connect to other nodes within their community than outside their community).

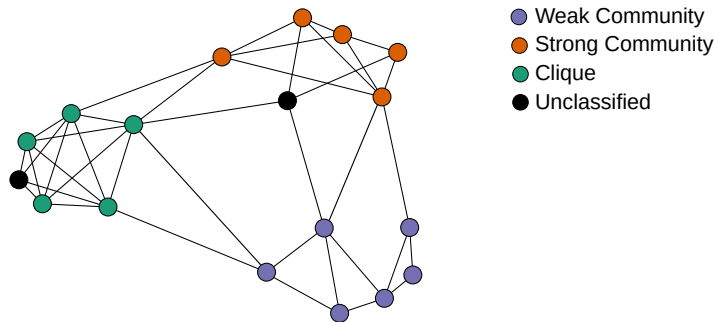
The most basic possible definition of a community which satisfies these properties is a complete subgraph, or *clique*; a subgraph  $C$  of the main graph  $G$  in which every node is connected to every other node (see Figure 2.8). However, this definition proves too rigid for practical use. Above the level of triangles, there are very few cliques within most networks, and many plausible community structures can be devised without enforcing completeness, such as below.

For a given connected subgraph  $S$  of the graph  $G$ , and a given node within that subgraph,  $i$ , we can distinguish between the *internal degree*,  $k_i^{int}$ , the number of edges connecting the node to other nodes in  $S$ , and the *external degree*,  $k_i^{ext}$ , the number of edges connecting the node to nodes not in  $S$ . Two definitions of communities can then be given:

*Strong community*: a subgraph in which for each node  $i \in S$ ,  $k_i^{int}(S) > k_i^{ext}(S)$ .

*Weak community*: a subgraph in which  $\sum_{i \in S} k_i^{int}(S) > \sum_{i \in S} k_i^{ext}(S)$ .

A weak community provides a less severe constraint on the possible nodes within the community, as nodes can have more edges leaving the community than remaining in it, as long as the total internal degree of the community remains higher than the total external degree.



**Figure 2.8:** Examples of three possible community definitions: a clique (green), a strong community (orange), and a weak community (purple). The two black nodes are unclassified. This demonstrates that the clique is most likely too rigid a unit to use as a community definition.

A better definition would move away from edge counting to focus on the probability of an edge occurring inside or outside a community. However, the problem of calculating the edge probabilities remains in general ill-defined [6].

Community detection is a challenge for three reasons:

1. Perhaps most importantly, there is no clear definition of what constitutes a community, let alone what metric of “goodness” a community should obey. This reflects the many different requirements on the communities found, and is highlighted by the wide range of terms used to describe communities (e.g. blocks, modules, clusters).
2. Unlike the related problem of graph partitioning [78], the number of communities, if any, is not known in advance.
3. The total number of partitions for a system rises faster than exponentially with the number of nodes (see Appendix A.1), even in the simplest case of network bisection. This makes exhaustive search of partitions computationally unfeasible.

## Community detection methods

There have been a multitude of algorithms developed for identifying communities within networks. Often these algorithms are developed to generate communities with different properties. Many attempts have been made to group, classify and evaluate these methods. Schaub [7] uses a problem-based approach, in which the motivation for finding the communities can be split into four categories:

- **Cut-based:** Communities are generated to minimise the external links, such as in the Kernighan-Lin algorithm [79], regardless of any internal structure.
- **Clustering-based:** Communities are required to be densely connected, as in data clustering.
- **Inference-based:** Here the detection process aims to identify groups of structurally equivalent nodes.
- **Dynamic:** Here the communities are used to simplify the description of processes occurring in the network.

Community detection algorithms are traditionally specialised towards one of these requirements. In addition, there have been many attempts to compare the accuracy of methods, based upon how well they reconstruct known communities [80–85]. These communities are either known from metadata collected about the network, or artificially generated (see Section 2.4). It has recently been shown that no method can perform best on all networks [86]; specifically, that the average accuracy of any method  $f$  over all possible community detection problems is a constant, independent of  $f$ . This “No Free



Lunch” theorem implies that, as in unsupervised learning, the method must be selected to address the specific problem at hand.

To show the diversity of approaches taken to detecting community structure, four popular methods are given below.

### **Spectral Methods:**

These make use of the properties of a matrix associated with the graph; most usually, the Laplacian. The Laplacian  $L$  has elements [53]:

$$L_{ij} = \begin{cases} k_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \exists \text{ an edge } (i,j). \\ 0 & \text{otherwise.} \end{cases}$$

Here  $k_i$  is the degree of node  $i$ . It can be shown that the eigenvectors of this Laplacian give key information on the connectivity of the nodes in the graph; this can be exploited to cluster the nodes. This is closely related to the problem of minimum cut. However, spectral methods often fail when the graph is sparse (as is often the case in real-world networks), and require the number of communities to be specified in advance [87].

### **Modularity optimisation:**

Perhaps the most popular method used in community detection, this technique relies on the fact that a random graph is not expected to exhibit community structure. By comparing the connectivity within a community to that of a null model, we can define a metric of community “goodness” that can be optimised to find the best community structure in the graph [88]. The modularity is defined as:

$$Q = \frac{1}{2w} \sum_{i,j} (A_{ij} - P_{ij}) \delta(C_i, C_j)$$

Where  $N$  is the number of nodes, with  $i, j \in [1, N]$  and  $\delta(C_i, C_j) = 1$  if nodes  $i$  and  $j$  belong to the same community, and 0 otherwise.  $A_{ij}$  and  $P_{ij}$  are the adjacency matrices of the network and of the null model, respectively.  $w$  is the total number of edges in the network. Choosing that the null model has the same degree distribution as the network under study gives:

$$Q = \frac{1}{2w} \sum_{i,j} \left( A_{ij} - \frac{A_i A_j}{2w} \right) \delta(C_i, C_j)$$

This recasts the problem of community detection as an optimisation problem, which allows existing optimisation methods to be applied (though modularity maximisation is NP-hard [89], meaning approximate algorithms must be used). However, modularity optimisation suffers from two key flaws. Firstly, the modularity has been shown to have a resolution limit [90]. If the communities within a graph are smaller than a certain size, relative to the total size of the graph, then they will not be found. (It has been shown that this limit applies in general to any method with a comparison to a null model [91, 92].) Secondly, the modularity is highly degenerate [93]; the number of local maxima with modularity close to the global maximum increases exponentially with system size. This reduces the confidence in the resulting partition.

Interesting, modularity maximisation can be interpreted as both maximisation of a specific type of stochastic block model [94], and of a specific diffusion process on the network [95].

### Stochastic Block Models:

The stochastic block model (SBM) is a generative network process which creates a random graph containing some known community structure [71]. Specifically, the model takes a partition of the nodes into  $b$  communities, and a  $b$ -by- $b$  matrix listing the number of edges linking each community; edges are then placed randomly satisfying these constraints. This model assumes that edges are placed randomly inside each community, which results in nodes within the same community having very similar degrees. As many real-world networks have highly varying degrees, the *degree-corrected stochastic block model* corrects for this by specifying the degree sequence of each node as an additional parameter [96].

Once we have a generative process, Bayesian inference can be used to obtain the probability that the network under study would be seen as a result of this process. By finding a block model that maximises this probability, we obtain the community structure for the network [97].

This method has the advantage that the nature of the inference process automatically provides the significance of the community structure, e.g. as a p-value. On the other hand, this method also suffers from a resolution limit, scaling as  $\mathcal{O}(\sqrt{N})$ , as in the case of modularity maximisation; nested models can reduce this limit to  $\mathcal{O}(\log N)$  [98].

**Infomap:**

Infomap uses the equivalence between compression of data and finding repeated patterns within that data (known as Minimum Description Length statistics [99]), where the data to be compressed is here the dynamics of a random walker moving through the network.

Each node is assigned a label. The movement can then be described by specifying the sequence of nodes visited by the walker. However, we expect that if the network has strong communities, the random walker will spend most time within a community, transitioning between communities infrequently. As such, using a second set of labels to describe the communities, then reusing the node labels, will reduce the information required to describe the movement, also known as the *description length* (e.g. rather than node 1, node 147, node 80 we could say community 1, node 1, community 1, node 2, community 1, node 3 etc.). It can be shown that Shannon's source coding theorem [100] gives the per-step description length as [101]:

$$\mathcal{L}(M) = q_{\cap} H(Q) + \sum_{i=1}^m p_i H(P_i)$$

- $\mathcal{L}(M)$  is the per-step description length for the partition M.
- $q_{\cap}$  is the rate at which the index reference is used.
- $H(Q)$  is the average length of the labels in the index reference, weighted by frequency.
- $p_i$  is the rate at which the  $i^{\text{th}}$  community reference is used.
- $H(P_i)$  is the average length of the labels in the  $i^{\text{th}}$  community reference, weighted by frequency.

Assigning communities to minimise this description length will reveal the natural community structure of the network. This method has the advantage that its resolution limit depends on the total weight of edges between communities, rather than the weight of all edges in the network [102]. This allows it to find smaller communities in larger networks. The method is recursive, finding nested community structure in a single pass of the algorithm [103].

The Infomap algorithm is used to generate the communities given in Chapter 4 and Chapter 5.

## Validating Community Structure

The four example methods given in Section 2.4 highlight the diversity of approaches taken towards community detection. In order to compare these diverse methods, we need a way to validate the output partition; this requires metrics of community “goodness”. We can distinguish between structural metrics, which measure the properties of the communities as locally dense subgraphs, and performance metrics, which measure how well the community structure agrees with benchmark data [82].

### Structural Metrics

Structural metrics are based upon measures of internal connectivity within the community, external connectivity from the community to the rest of the graph, or some combined measure of the two. For a graph within  $N$  nodes,  $N_{\mathcal{S}}$  of which are in the community  $\mathcal{S}$ , example metrics are (see also [104]):

- Internal density: the fraction of possible edges within the community that actually occur.  $\sum_{i \in \mathcal{S}} k_i^{int}(\mathcal{S}) / N_{\mathcal{S}}(N_{\mathcal{S}} - 1)$
- Cut ratio: the fraction of all possible edges leaving the community,  $\sum_{i \in \mathcal{S}} k_i^{ext}(\mathcal{S}) / N(N - N_{\mathcal{S}})$ .
- Conductance: the ratio between the number of edges leaving the community, and the number of edges within the community:  $\phi(\mathcal{S})$  for a community  $\mathcal{S}$  is given by  $\sum_{i \in \mathcal{S}} k_i^{ext}(\mathcal{S}) / \min(\sum_{i \in \mathcal{S}} k_i^{int}(\mathcal{S}), \sum_{i \in \bar{\mathcal{S}}} k_i^{int}(\bar{\mathcal{S}}))$ . Here  $\bar{\mathcal{S}}$  is the set of nodes not in community  $\mathcal{S}$ . Lower conductance implies a more well-defined community, and can be interpreted in terms of “surface-area-to-volume ratio”.

The modularity can also be used, as a measure of community non-randomness. As expected, many of these measures are highly correlated.

### Performance Metrics

Measuring how well a community detection algorithm can find known community structure requires a reliable set of benchmark networks. These benchmark networks can either be real-world networks with known community structure (although the level of trust in metadata as a proxy for communities has been recently brought into question [86]), or between artificially generated benchmark networks. The most widely used benchmark is that of Lancichinetti, Fortunato and Radicchi (LFR) [105]. This will generate networks of

an arbitrary size, and with an arbitrary number of communities. The degree distribution and community size distribution obey power laws with distinct exponents, reflecting commonly observed community structures, and with a mixing parameter  $\mu$  which gives a variable level of random network rewiring.

Performance metrics measure the similarity between a generated and an expected partition. They can be split into three general classes (following [6]); those based on pair counting, those based on cluster overlap, and those derived from information theory. Examples of each are given below.

**Rand index:** For all pairs of vertices across the two partitions, we identify those pairs classified as belonging to the same community, and those belonging to different communities. Define the number of pairs assigned to the same community in both partitions as  $\mathcal{A}$ , the number assigned to the same community in one partition and different communities in the other as  $\mathcal{B}$  and  $\mathcal{C}$ , and the number of pairs assigned to different communities in both partitions as  $\mathcal{D}$ . Then the Rand index between partitions  $\mathcal{X}$  and  $\mathcal{Y}$  is defined as:

$$R(\mathcal{X}, \mathcal{Y}) = \frac{\mathcal{A} + \mathcal{D}}{\mathcal{A} + \mathcal{B} + \mathcal{C} + \mathcal{D}}$$

i.e. the fraction of pairs that are correctly classified.

**Jaccard index:** Using the same terminology, the Jaccard can be defined as:

$$J(\mathcal{X}, \mathcal{Y}) = \frac{\mathcal{A}}{\mathcal{A} + \mathcal{B} + \mathcal{C}}$$

i.e. the number of pairs classified into the same community in both partitions, divided by the total number of pairs in which at least one partition groups them together. The Jaccard can also be formulated more generally as the ratio of the intersection and the union of two sets, where here the sets are of vertex pairs classified into the same community in two different partitions.

$$J(\mathcal{X}, \mathcal{Y}) = \frac{|\mathcal{X} \cap \mathcal{Y}|}{|\mathcal{X} \cup \mathcal{Y}|}$$

Neither the Jaccard nor the Rand indices take the full range of values from 0 to 1, however standard z-scoring can be used to assess the significance of the result.

**Fraction of correctly detected vertices:** A vertex is defined as correctly detected if more than half the nodes in the generated community are assigned to the same expected community, with the caveat that if two or more expected communities fit within the generated community, all nodes within it are rejected as misclassified. The number of correctly detected vertices divided by the total number of nodes then gives a similarity measure.

**Normalised Mutual Information (NMI):** This is perhaps the most widely used similarity measure, evaluating the amount of information shared between the two partitions (derivation follows [23]). The Shannon Entropy,  $H_A$ , for a partition  $\mathcal{X}$  is defined as:

$$H_A = - \sum_{i=1}^{m_X} \frac{N_i}{N} \log \frac{N_i}{N}$$

Where  $m_X$  is the number of communities in partition  $\mathcal{X}$ ,  $N_i$  is the number of nodes in community  $i$ , and  $N$  is the total number of nodes. The mutual information is defined in terms of a “confusion matrix”, where the element  $N_{ij}$  gives the number of nodes in community  $i$  of partition  $\mathcal{X}$  that are also in community  $j$  of partition  $\mathcal{Y}$ :

$$I(A, B) = \sum_i \sum_j \frac{N_{ij}}{N} \log \left( \frac{N_{ij} N}{N_i N_j} \right)$$

Finally, normalising by the total Shannon Entropy gives the normalised mutual information  $I_N$  between the two partitions  $\mathcal{X}$  and  $\mathcal{Y}$ :

$$I_N(A, B) = \frac{2I(A, B)}{H(A) + H(B)}$$

This gives a performance metric such that  $0 \leq I_N \leq 1$ , with 0 indicating zero agreement and 1 perfect agreement between the two partitions.

In this work, we use a modified version of the Jaccard index for assessing performance, and the conductance for assessing the structure of the communities.

## 2.5 Previous work on protein structure networks

It is the complex pattern of bonding in the protein that determines its shape and function. A network description seems a feasible route for exploring this pattern of bonding, and has indeed become a popular avenue for investigating protein topology. Surveying the literature is made more complicated by the popularity of protein interaction networks, in which whole proteins are the nodes, linked if they interact either genetically or physically in the cell. These have found extensive use in the mapping of the proteome [106–108].

Most investigation of protein structure networks has been done using Residue Interaction Networks (also referred to as Amino Acid Networks or Protein Contact Networks), in which the residues are the nodes, and distance thresholding is used to create edges [109–111]. These networks have been used to identify key residues within each protein, such as for protein stability or allostery [12–14]. Web servers [112] and network visualisation tools [113,114] have been developed to aid this analysis.

In terms of the application of community detection to protein structure, existing efforts are surveyed below.

- Barahona et al. have demonstrated the mapping of communities to functional domains for the case of myosin and adenylate kinase, using atomistic networks generated using bonding calculations [115,116].
- Tasdighian et al. have compared the results obtained from traditionally k-means spatial clustering with that obtained from community detection on a set of 11 protein structures, and shown that the network model gives more promising results [117].
- Feldman has used spatial clustering methods to identify structural domains in  $\sim 7000$  protein chains, and shown that these domains broadly agree ( $\sim 80\%$ ) with the SCOP and CATH databases [118].
- Hleap et al. have used correlation networks, in which residues are linked if they are related by homology, to identify sub-domain level architecture within a set of 24  $\alpha$ -amylase proteins [119].
- Finally, Csermely et al. have developed a plugin for Cytoscape, a widely used visualisation tool, to enable the automatic generation and visualisation of the overlapping community structure resulting from a residue interaction network [120].

Existing efforts differ from the work presented here in key ways. No previous work has systematically analysed the community structures of large sets of proteins, either

as a method for generating candidate protein domains (Chapter 4) or as a tool for investigating protein topology (Chapter 5).



# Chapter 3

## Codebase development

In order to explore community detection as a tool for protein structure analysis, we need an automated way to convert solved protein structures into networks, run community detection on the networks, and store, analyse, and visualise the results. This process involves a large number of design choices and hyperparameters, which may greatly influence the results. In this chapter we introduce the computational tools developed, and explain the rationale behind their design.

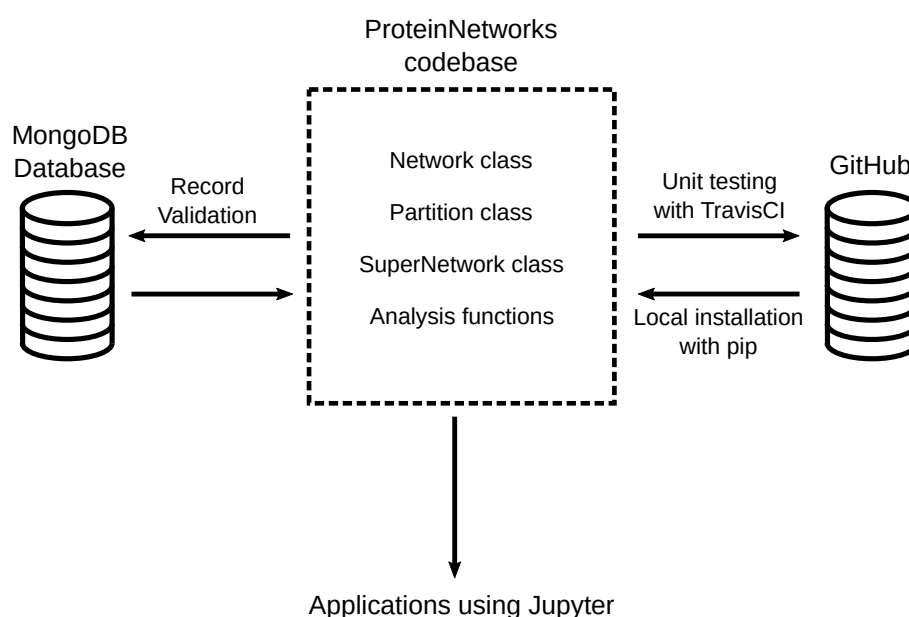
### 3.1 Software architecture

The software was written in Python. This language has wide use in the bioinformatics and network analysis communities, with a full-featured testing, packaging and documentation system, and a language design that allows for quick exploratory coding. We can use standard packaging tools for installation. This improves reproducibility and reduces the initial effort needed for future users. Each section of the pipeline is an encapsulated submodule, with error checking at all stages. For example, the network generation process is encapsulated in a `Network` object, with rigorous input and output validation. This allows users to have confidence in the results obtained. A suite of automated tests can be run using `pytest`, and the regions of the code currently being tested can be checked using `coverage.py`. Again, this use of industry-standard monitoring allows for confidence in the results. The code is stored on GitHub, and the full test suite is run on every update to the GitHub repository using TravisCI to monitor for new errors.

This Python package is connected to a secured MongoDB database storing the generated networks and community structures, preventing redundant computational effort. If a network with a given set of parameters is requested by the user, first the database is

checked. If a matching network is found, it is retrieved. Otherwise, a new network is generated and stored. In this way, a large dataset of structural data, network edge lists, and community structures gradually accrues, without duplication of effort.

Care was taken to distinguish between the user and developer of the codebase - the applications of the code given above, i.e. the code used to generate actual results, are stored in a separate repository, and are generally written as Jupyter notebooks [121]. The full codebase is available at <https://github.com/ProteinNetworks/>.



**Figure 3.1:** Logic diagram for the codebase. Arrows indicate the direction of the flow of data.

The three major components of the codebase are network generation (handled by the Network class), community detection (handled by the Partition class), and output storage/analysis. Network generation and community detection will be covered in this chapter, whilst the analysis tools will be introduced as required in subsequent chapters.

## 3.2 Network generation

The central data store of protein structure is the Protein Data Bank (PDB), a publicly accessible repository in which each solved structure is indexed by a four-character reference. Each structure is stored as a .pdb file. This file format is rigorously defined, with each row specified by a given descriptor, and every column given a specific role (see

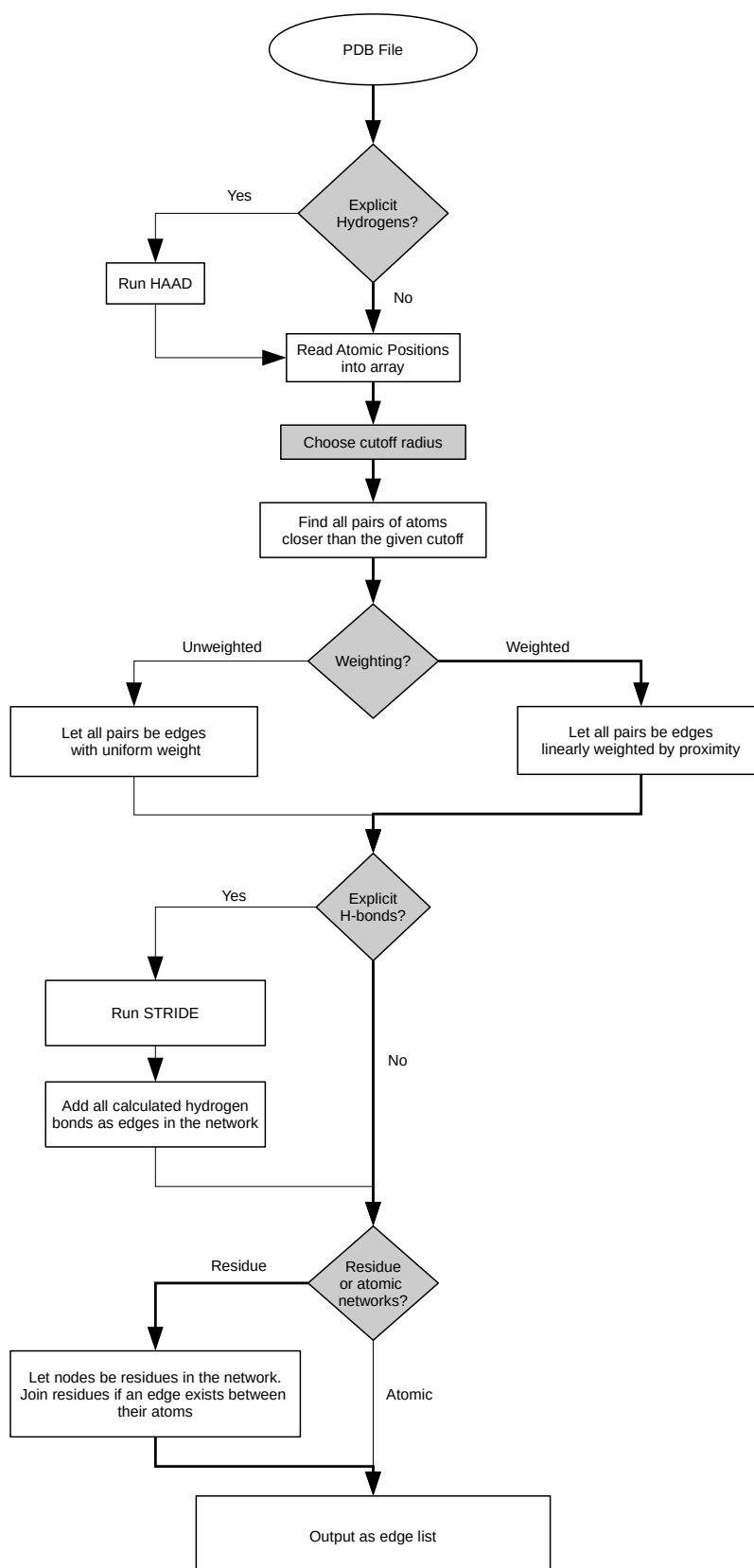
Figure 3.2). This allows for easy fetching and parsing of the data. In order to apply community detection methods to the protein structure, we must first convert this list of atomic positions into a network representation.

ATOM	13	CB	GLN	A	2	26.733	30.148	2.905	1.00	14.46	C
ATOM	14	CG	GLN	A	2	26.882	31.546	3.409	1.00	17.01	C
ATOM	15	CD	GLN	A	2	26.786	32.562	2.270	1.00	20.10	C
ATOM	16	OE1	GLN	A	2	27.783	33.160	1.870	1.00	21.89	O
ATOM	17	NE2	GLN	A	2	25.562	32.733	1.806	1.00	19.49	N
ATOM	18	N	ILE	A	3	26.849	29.656	6.217	1.00	5.87	N
ATOM	19	CA	ILE	A	3	26.235	30.058	7.497	1.00	5.07	C
ATOM	20	C	ILE	A	3	26.882	31.428	7.862	1.00	4.01	C
ATOM	21	O	ILE	A	3	27.906	31.711	7.264	1.00	4.61	O
ATOM	22	CB	ILE	A	3	26.344	29.050	8.645	1.00	6.55	C
ATOM	23	CG1	ILE	A	3	27.810	28.748	8.999	1.00	4.72	C

**Figure 3.2:** Example rows from a PDB file - this is the protein with reference 1UBQ, ubiquitin. Every row starts with a row identifier - here ATOM indicates that what follows are the 3D co-ordinates of an atom in the structure. The atom's serial number, name, and element are specified, as well as the residue name and sequence number to which the atom belongs. The set of five decimals indicates the [x,y,z] position, the occupancy, and B-factor in order.

The network generation process implemented in this work involves reading the atomic positions from the PDB. The atoms of the protein are then used as nodes in the network, with an edge between nodes if the corresponding atoms are within a certain distance. This process has the following key parameters (shown in grey on Figure 3.3 overleaf, and described in detail in what follows).

- Treatment of hydrogen atoms.
- Distance threshold used.
- Weighted vs unweighted networks.
- Residue networks vs atomic networks.



**Figure 3.3:** Logic diagram for the network generation process. The route chosen for this work is shown in bold.

## Treatment of hydrogen atoms

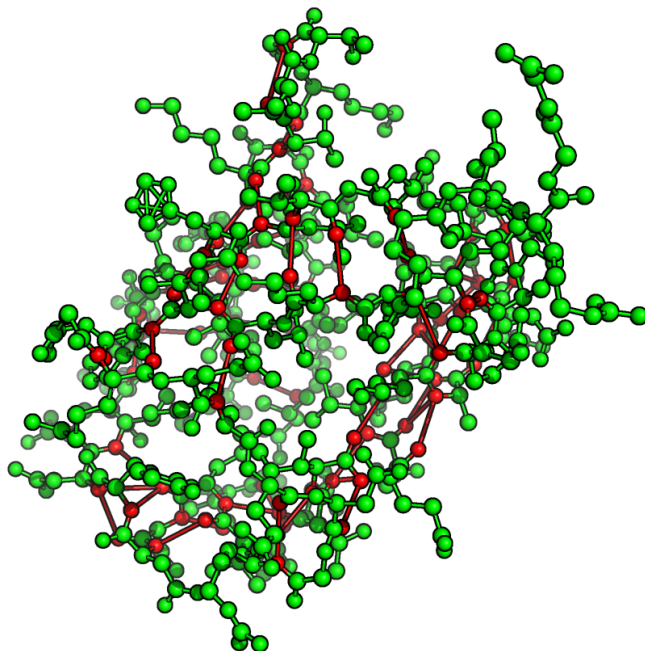
The protein structures stored in the PDB have overwhelmingly been solved using X-ray diffraction (>90%, with the remainder solved using NMR and cryo-EM). Due to hydrogen's low electron density, this method is unable to identify the positions of the hydrogen atoms directly, and as such the raw PDB data contains no information on hydrogen positions.

If the goal of the network generation process is to accurately represent the pattern of bonding within the protein, this exclusion represents a challenge. The C-C covalent bond length is  $\sim 1.5\text{\AA}$ , whilst the average hydrogen bond length (donor-acceptor distance) is  $\sim 3\text{\AA}$ . As such, any distance-threshold based approach faces a choice between excluding hydrogen bonds (known to be key for proper protein folding) and introducing unphysical next-nearest-neighbour covalent bonds. There are two possible approaches to rectify this.

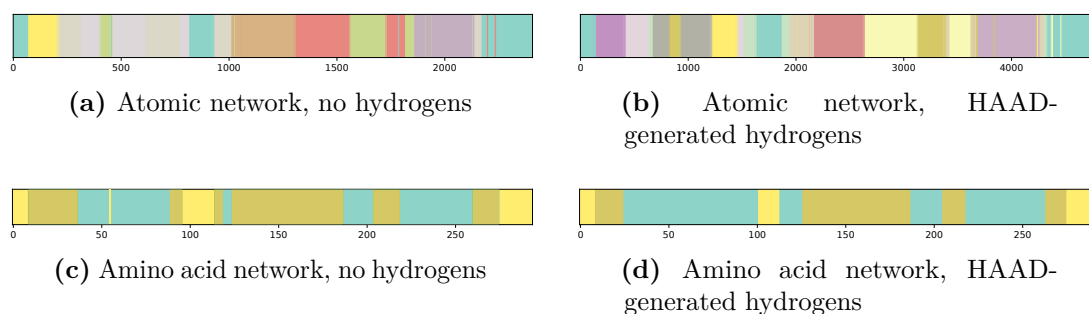
1. **Add hydrogens at their predicted positions.** Whilst the hydrogen atoms are not explicitly included, the amino acid sequence of a protein is known, and the amino acid structure remains largely invariant. This means that, in principle, the location of the hydrogen atoms can be predicted, from constraints on the valency and bond lengths of the surrounding atoms. The program HAAD [122] can be used to process the PDB file and predict the location of each hydrogen. This then reduces the distance threshold required to capture hydrogen bonding.
2. **Calculate the likely hydrogen bonds, and add these as edges.** Hydrogen bonds are highly directional, and tools exist to predict their occurrence. These tools are integrated into most molecular viewers, but here the program STRIDE is used [123]. This allows a low distance threshold to be used to capture the covalent bonding within the chain, and edges representing the hydrogen bonds to be added afterwards (see Figure 3.4 for an example of the bonds introduced by STRIDE).

Each of the above tools introduce external dependencies into the code, and vastly increase the time required for network generation. The HAAD program in particular will occasionally crash with a SIGSEGV error, making incorporation into the codebase difficult. Both codes add hydrogen-based information by altering the PDB file in a manner that makes systematic parsing more difficult. In addition, when comparing the community structures of networks with and without hydrogen atoms, we see no gross difference in the resulting communities found. This holds both for atomic networks and for amino acid networks (see Figure 3.5). As such, we elected to ignore hydrogen atoms

in the network generation process, instead choosing a distance threshold that would incorporate the relevant information (see overleaf).



**Figure 3.4:** An example of the edges generated using STRIDE on ubiquitin (PDB reference 1UBQ); regular covalent bonds, shown in green, are found using a cutoff scaling of 2. Hydrogen bonds are calculated using the STRIDE program, shown in red.



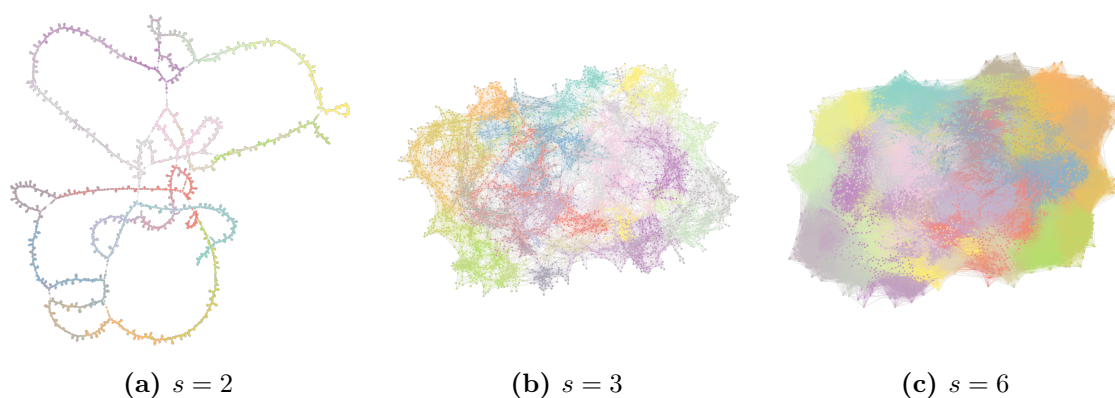
**Figure 3.5:** Comparison between the community structure generated before and after hydrogen addition using HAAD [122], for atomic networks (top) and amino acid networks (bottom), using the Infomap community detection algorithm [103], for the protein with PDB reference 3P8T. The community structure is plotted as a linear sequence, with distinct colours corresponding to distinct communities. We see that overall the community structure does not significantly alter as a result of the hydrogen addition.

### Distance threshold used

Edges are generated between nodes if the corresponding atoms are within a certain distance. Clearly this distance is a key factor in the connectivity of the resulting network. When the distance chosen is less than the maximum covalent bond length, the graph will no longer be connected and community detection will fail. At the opposite extreme, a fully connected ((N-1)-regular) graph will also fail to give relevant communities. The distance threshold giving the best communities is unknown; however, previous literature [110,112] has suggested a cutoff distance of between 8Å and 5Å for networks of amino acids. Whilst the most simple choice is an absolute cutoff, this fails to take account of the variable atom size. As such, the distance threshold between atoms  $i$  and  $j$  is defined as:

$$c_{ij} = s (r_i + r_j)$$

Where  $r_i$  is the covalent radius of atom  $i$  (as defined in [124]), and  $s$  is a scaling. It is this scaling that will be varied as a free parameter in the codebase, with  $s = 4$  being used in Chapter 4 and Chapter 5.



**Figure 3.6:** Examples of high and low scaling parameters, for protein 3P8T. Networks are plotted using the SFDP spring-layout algorithm [125], as part of the graph-tool package [126]. These plots demonstrate that strong dependence of the scaling on the graph properties.

### Weighted vs unweighted

Networks can be weighted or unweighted (see Section 2.3), and the choice will impact the community detection algorithm used, as many cannot handle weighted networks.

Unweighted networks are more space-efficient, and algorithms generally run more quickly. However, by weighting the network we can encode more information about the nature of the bonds. As bond energy and bond length negatively correlate [127], we choose the following linear weighting:

$$A_{ij} = \begin{cases} \frac{c_{ij}-|d_{ij}|}{c_{ij}} & \forall d_{ij} \leq c_{ij} \\ 0 & \text{otherwise} \end{cases}$$

where  $A_{ij}$  is the weight of the generated edge, and  $|d_{ij}|$  is the absolute distance between atoms  $i$  and  $j$ .  $c_{ij}$  is the cutoff distance as defined previously. This linear weighting scheme should induce community detection algorithms to preferentially break weaker bonds over stronger bonds.

### Amino acid networks vs atomic networks

Amino acid networks have been used previously in the literature to investigate the structure of proteins [110,112]. For an amino acid network, the edges are generated if two residues are within a certain distance. This distance measure can be based upon the inter- $C_\alpha$  distance, or on the number of pairs of atoms within a certain proximity. Here the latter is used.

However, these networks will inevitably carry less information on the protein's true bonding (for example, due to side-chain movement), than an atomic network. If the amino acid network can capture all the relevant features of the protein, then it should be used, as this resulting network will be significantly smaller. These will result in quicker analysis, and a faster development time.

As shown overleaf, we see no structural distinction between amino acid networks and atomic networks, and as such amino acid networks are used throughout this work.

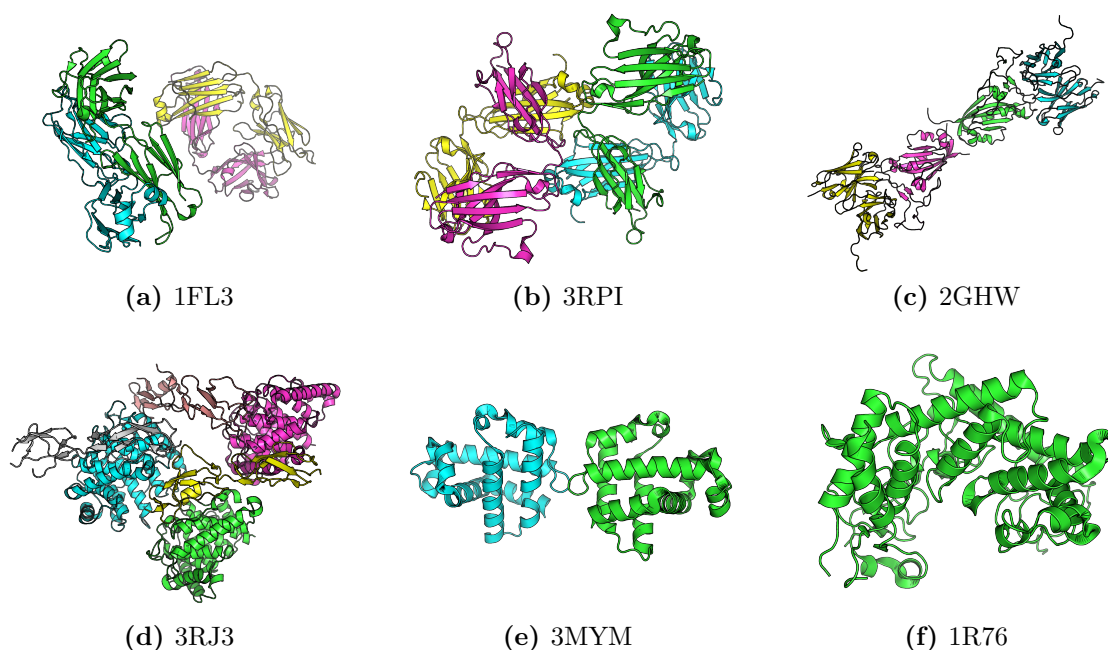
Performing this analysis on a protein with multiple chains often results in a network with distinct connected components, corresponding to each chain. As such, for this analysis the proteins are first split by chain. This helps ensure that any results are fixed at the sub-quaternary level.



## Testing the network parameters

The effect of the input parameters chosen on the network generated can be investigated directly by plotting the network's properties as a function of the input parameters, for a selection of test proteins:

PDB Reference	PFAM ID	SCOP ID	Protein Size / Atoms
1FL3	PF07686.13	b.1.1.2	6364
3RPI	PF07686.13	b.1.1.0	6476
2GHW	PF07686.13	d.318.1.1	6614
3RJ3	PF00084.16	a.102.4.4	9840
3MYM	None	a.1.1.2	2586
1R76	PF09492	a.102.5.1	2963



**Figure 3.7:** The test proteins used in investigating the properties of the generated networks, indexed by PDB accession code and colored by chain.

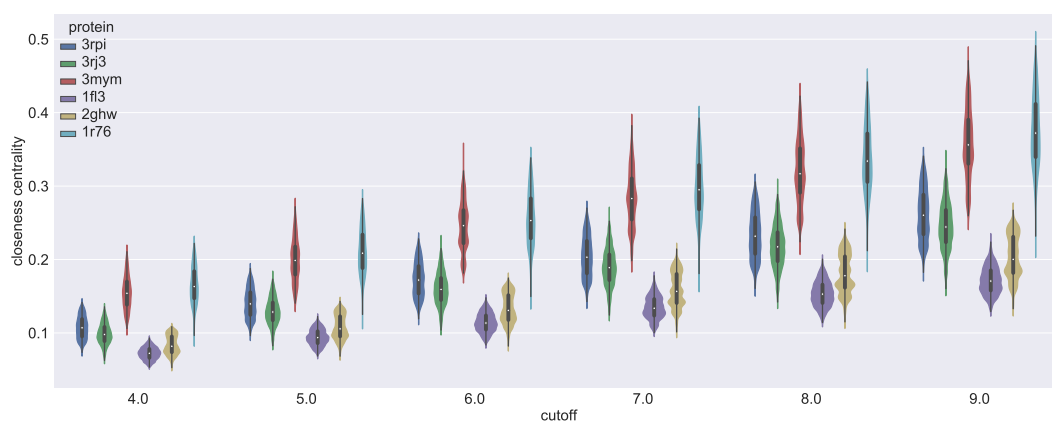
These proteins are chosen to exhibit a range of sequences, structures and sizes. The aim is to generate community structures that are as informative as possible about the protein's underlying structure. As such, the network properties of structurally distinct

proteins should be as distinct as possible. Two network measures are explored; closeness centrality, and the clustering coefficient.

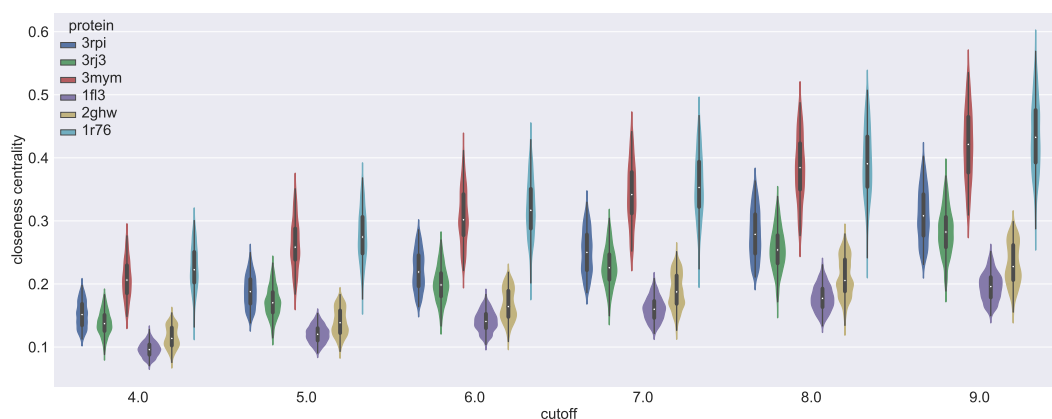
**Closeness Centrality:**

The closeness centrality is a measure of how central a node is within a network, as the average of the shortest path lengths between a node and all others, normalised by the graph size. Figure 3.8 gives the distribution of the closeness centrality for each test protein, for several choices of scaling parameter  $s$ .

It can be seen that the distributions for the amino acid and atomic networks agree, giving confidence in the use of amino acid networks in further analysis. However, the distributions of closeness centrality do not discriminate between SCOP families, but rather between proteins of different size.



(a) Atomic networks

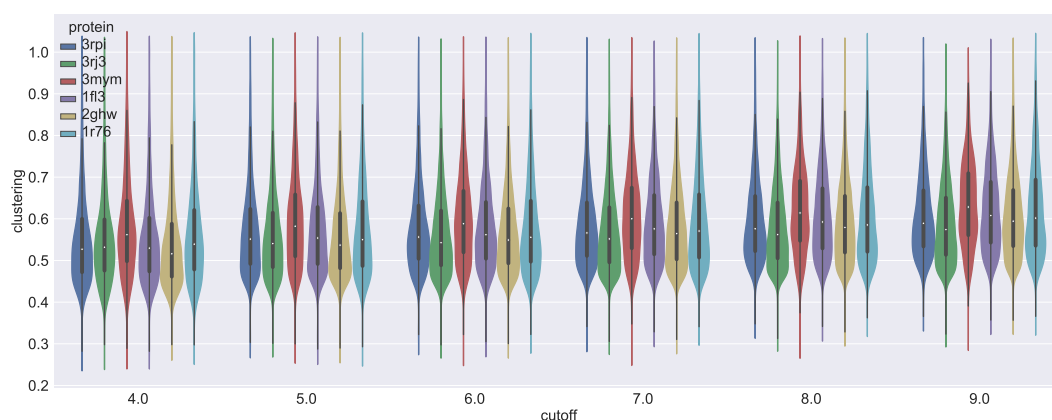


(b) Amino Acid networks

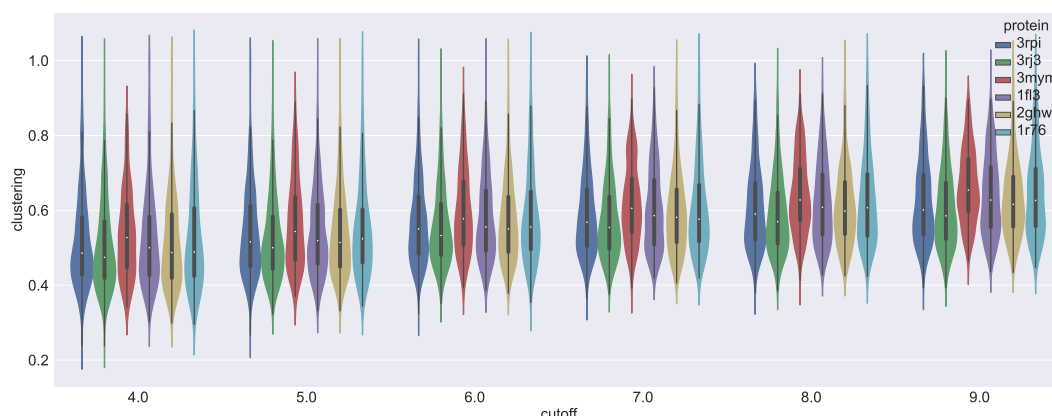
**Figure 3.8:** The distribution of closeness centralities for 6 test proteins, for different values of the distance threshold. Each coloured shape can be considered as a “vertical histogram”, showing the frequency density of the closeness centrality for a given protein and cutoff. Several conclusions can be drawn. Firstly, that the closeness centralities of the amino acid and atomic networks are almost indistinguishable. Secondly, increasing the cutoff radius alters the distributions only by an overall scaling. Thirdly, the SCOP membership (i.e. structural similarity) seems to have no bearing on the distribution. Proteins 3RPI (dark blue) and 1FL3 (purple) share a SCOP family, and proteins 3RJ3 (green) and 1R76 (light blue) share a SCOP superfamily, yet neither show similar distributions. Rather, there seems to be a strong agreement between the protein’s size and the breadth of its distribution.

### Clustering Coefficient:

The clustering coefficient, as defined in Section 2.3, gives the proportion of a node’s neighbours that are connected. The distributions of the clustering coefficient are given in Figure 3.9.



(a) Atomic networks



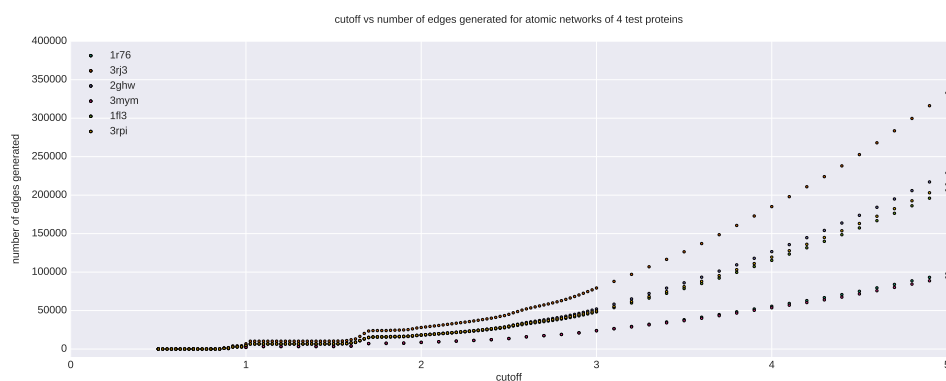
(b) Amino acid networks

**Figure 3.9:** The distribution of clustering coefficient for 6 test proteins, for different values of the distance threshold. As in Figure 3.8, the amino acid and atomic networks show excellent agreement, and the proteins' structural similarities are not reflected in the network plots. Unlike Figure 3.8, the broadness of the clustering distributions remains roughly constant with distance threshold, for values of the threshold used here.

As in Figure 3.8, the amino acid and atomic networks show strong agreement, and fail to discriminate between proteins with distinct topologies. These figures show that the traditionally measured properties of the networks cannot discriminate between structurally distinct and structurally similar proteins.

### Network Size:

We can also investigate the network size, i.e. the number of edges, as a function of the scaling parameter (Figure 3.10). As before, this appears to scale with the number of nodes in the network, as opposed to the structural class.

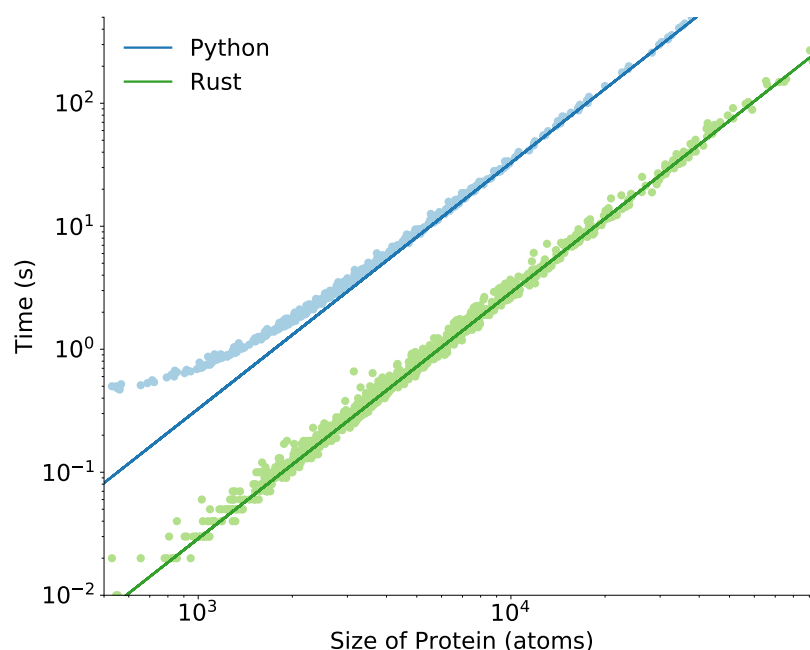


**Figure 3.10:** Variation of network size with cutoff radius for atomic networks.

Further traditional network properties are given in Appendix A.3.

## Performance

A network generator was implemented in Python with the options given above (except the treatment of hydrogen atoms). The all-to-all comparison gives the algorithm an  $N^2$  scaling, which makes the Python code too slow for analysis of the full PDB. As such, a Rust version of the code was implemented. This resulted in a code an order of magnitude faster (Figure 3.11).



**Figure 3.11:** The run time of Rust code vs Python code. Both show  $N^2$  scaling, but the Rust version lacks the fixed overhead of an interpreted language such as Python, so this scaling continues to extremely low times.

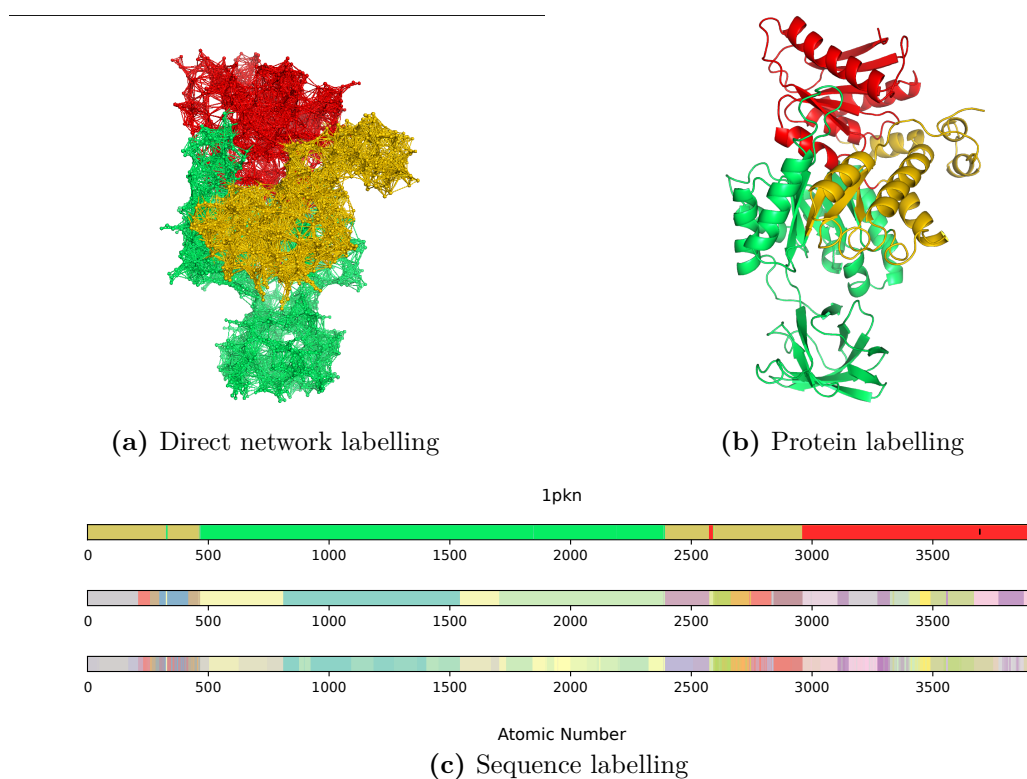
### 3.3 Community detection

Due to the wide variety of community detection algorithms in the public domain (see Section 2.3), no additional methods were implemented for this work. Most software for community detection takes the form of a compiled executable. This can then be integrated into the rest of the code. Development effort was focussed on ensuring a common interface for the outputs of different community detection methods.

Given a candidate community detection method, we can check the output either by examining the community structure on the network, on the sequence, or plotted directly onto the protein structure using Pymol [128]. Pymol is a 3D molecular visualisation tool written in Python. Pymol has powerful and flexible scripting for modifying or colouring structures, and reads a wide array of file formats, including PDB files.

All three visualisation methods have been implemented and integrated into the Python package (see Figure 3.12 for an example of direct network visualisation).

In choosing a community detection algorithm, we require a method that does not require the length scale or number of communities to be specified beforehand; we also require a method that is fast enough to allow for all 130 000 proteins in the PDB to be analysed in

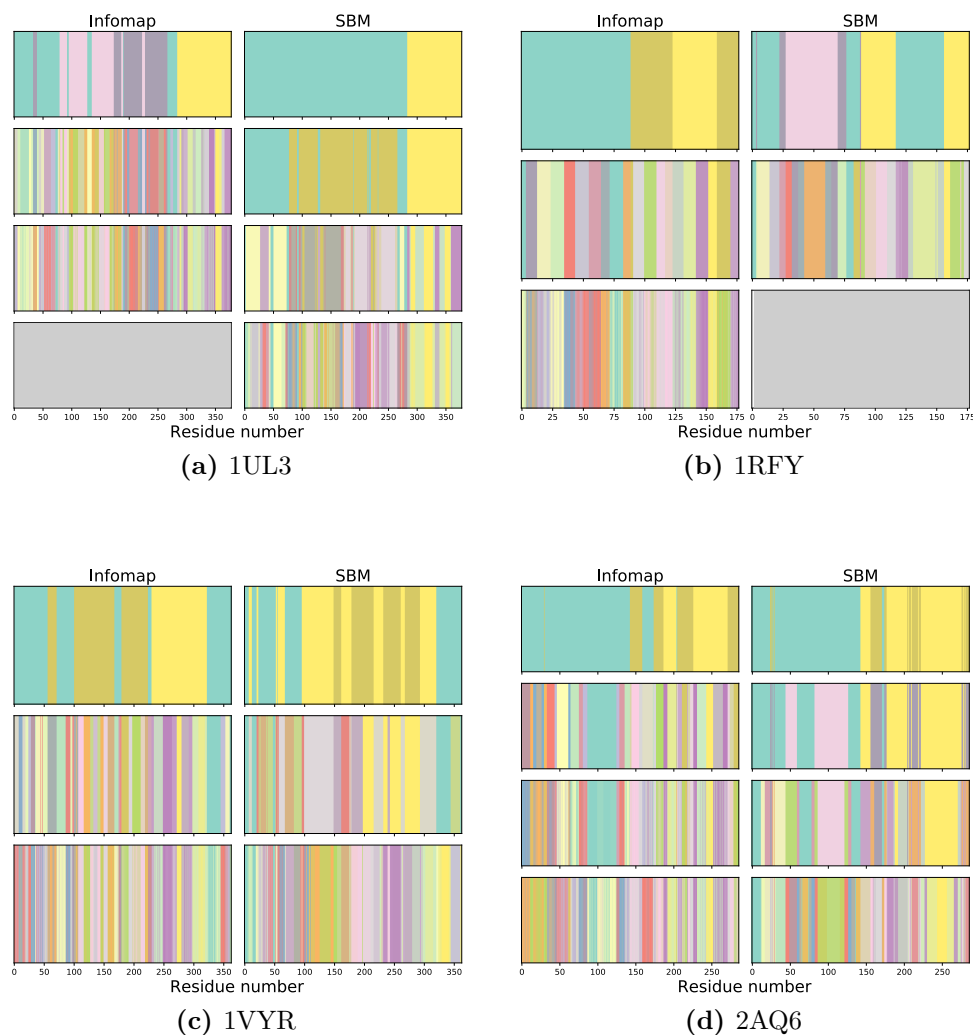


**Figure 3.12:** All three visualization methods exemplified on the 1PKN protein structure, on an atomic network with scaling parameter  $s = 3$  (a) Communities visualized on the network. (b) Communities visualized on the protein structure. (c) Communities at three levels mapped onto the linear amino-acid sequence.

a reasonable timeframe. We need the method to detect hierarchical community structure, in order to investigate the multi-scale structure of the protein, and a method with a resolution limit that will not impede the discovery of domain-level structure.

Three different community detection algorithms were selected as suitable, and investigated: **the AFG algorithm**, **Stochastic Block Models (SBM)**, and **Infomap**. A hierarchical SBM has been implemented as part of the open-source graph-tool package [126], and the Infomap algorithm has been open-sourced as a standalone executable under the GPLv3 license [103] (the relevant theory for both methods is covered in Section 2.4). The AFG algorithm, named after Arenas, Fernández and Gómez [129], has been implemented in previous MPhil work [23]. The AFG algorithm gave qualitatively correct results on test proteins (see Appendix A.2), but proved computationally unfeasible, requiring weeks or months of CPU time to analyse a single protein. The choice was therefore between SBM and Infomap. SBM and Infomap are fundamentally distinct community detection

methods, addressing different problems and finding communities according to different definitions.



**Figure 3.13:** Comparison between the hierarchical partitions generated using SBM (right) and Infomap (left), on the same protein networks, corresponding to the 4-digit PDB references given. The partitions are displayed with the most coarse-grained community structure on the top, with increasingly fine community below. When one method has generated more levels of community structure, grey sequences are displayed for the other. Broad agreement is demonstrated in the community structures of these four protein examples.

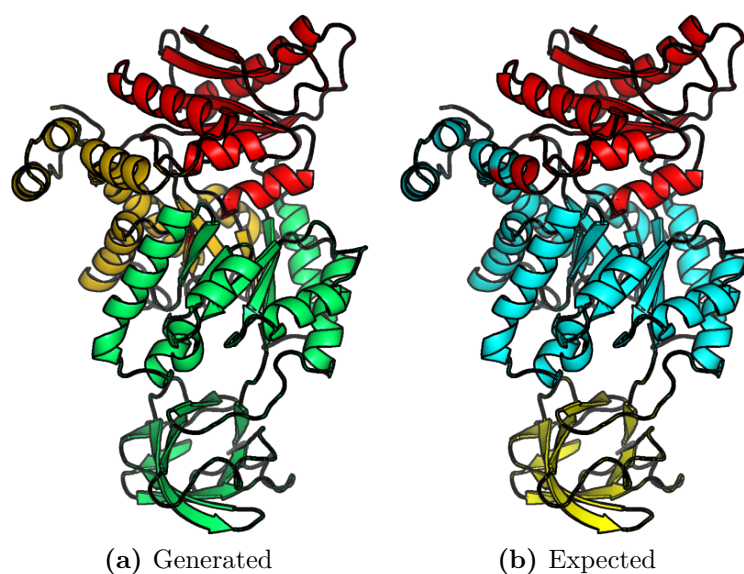
Hierarchical SBM is tightly embedded within the graph-tool package - the method works on graph-tool's own Graph data structures, and outputs a BlockState data structure. The graph-tool package itself is a highly optimised Python wrapper on C++ code which must be compiled. As it is based on Bayesian statistics, the output of the SBM process includes



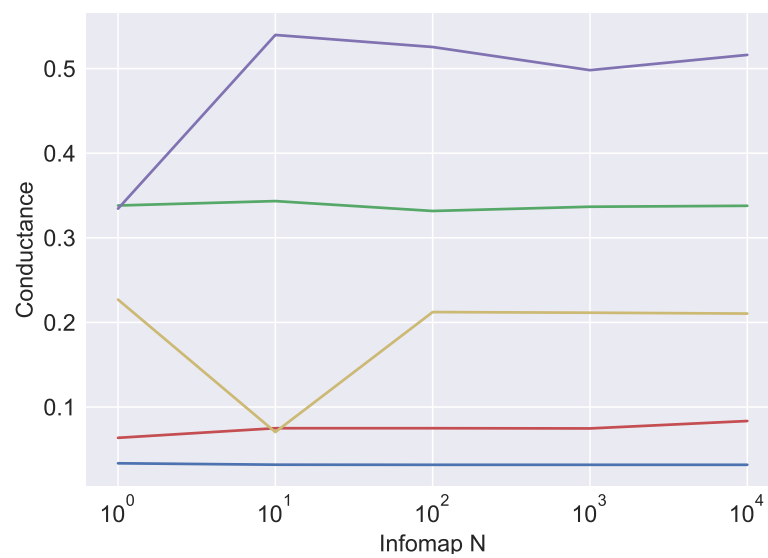
posterior probabilities of the community membership and the number of communities in total, but the method does not span the full protein structure (there are regions of the protein that will be unclassified).

Infomap exists as a stand-alone executable that could be easily embedded, with wider use in the literature, and generates a partition covering the full protein structure (which will be key in Chapter 5). Infomap has only one tunable parameter, which is the number of repetitions of the optimisation to carry out before choosing the best result.

Although Infomap and SBM broadly agreed when tested on 4 example proteins (Figure 3.13), Infomap was chosen for integration into the analysis pipeline, due to the ease of its implementation, and strict hierarchical structure. Infomap’s repetition parameter,  $N$ , was set to 1000, as this was the value at which the conductance (see Section 2.4) of the resulting communities converged (see Figure 3.15). Infomap has the disadvantage that it is prone to overpartitioning networks with geometric constraints, including spatial networks such as those generated in this work [130]. However, in this work, on networks generated with a scaling parameter of  $s = 4$ , this tendency is not pronounced (see Figure 3.14).



**Figure 3.14:** The generated community structure (a) using a residue network with scaling parameter  $s = 4$  compared with the known SCOP domains (b) for a pyruvate kinase with PDB code 1PKN. One colour signifies one domain/community. Here we see that one of the communities matches well to the existing SCOP domain (both shown in red). This figure is taken from [131].



**Figure 3.15:** The variation of conductance with infomap repetitions for five test proteins.

## Chapter 4

# Communities as conserved modules

Using the methodology developed in Chapter 3, we are now in the position to systematically fragment large sets of protein structures into substructures corresponding to the communities found on their networks. We expect these substructures to be compact, and highly intra-connected, therefore plausibly defining protein domains. One feature of domains is that they are conserved across different protein families. The extent to which the fragments found using this method are conserved, and how this process may inform the definition of novel candidate domains, will be the focus of this chapter.

We will first compare the derived communities with existing protein classification schemes based on sequence (Pfam) and structure (SCOP), as well as to previous work on protein structure networks. The method used here differs highly from existing approaches [119] each protein is processed individually, with no reference to previously found fragments, and without taking account of sequence similarity or evolutionary ties. Both the case where there is high overlap with existing methods, and the case where there is low overlap, are therefore interesting. Subsequently, we will present ways of analysing the derived communities in their own right.

### 4.1 Defining the modified Jaccard index

In order to make a comparison between Pfam or SCOP and our generated community structure, we first need to devise a similarity metric. The simplest method for comparing two protein structure annotations is to map both the community structure and the existing annotation onto the protein's sequence. The problem then becomes a straightforward comparison between two vectors of the same length. However, traditional performance metrics for comparing expected and generated community structure such as

the Normalised Mutual Information (see Section 2.4) are unsuitable for this task; the predicted structure (for instance the Pfam domain structure) occupies only a subset of the protein sequence, whilst the generated community structure assigns every residue to a community, thus tiling the structure completely. Additional assigned amino acids outside the region spanned by the conventionally predicted structure should not be penalised.

We therefore use a modified version of the Jaccard index for a comparison to the expected domain structure of each protein (the following analysis adapted from [131]). The Jaccard index is defined as the intersection between two sets, divided by their union, where in this case the sets correspond to regions of the protein sequence.

The index is modified as follows:

For each “expected” domain:

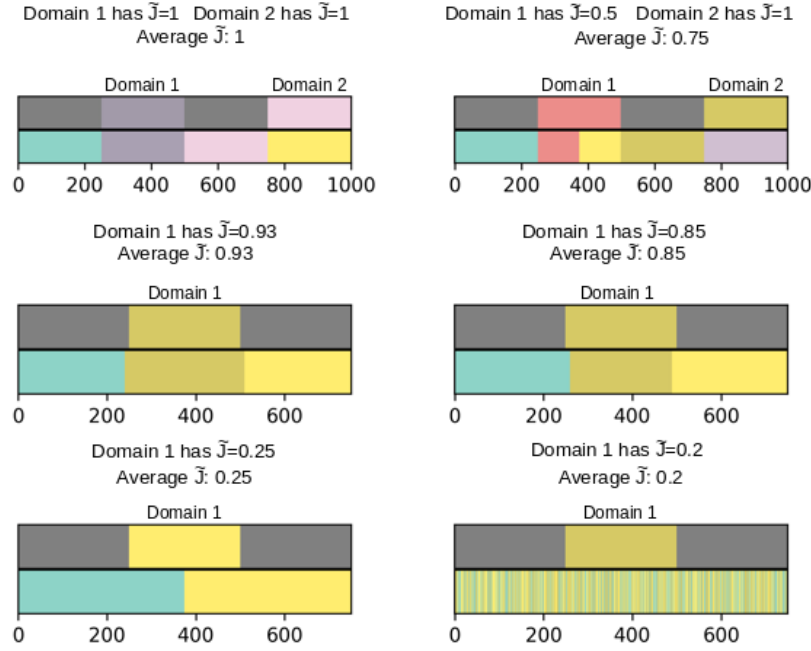
- Calculate the Jaccard index for all generated communities that overlap with the expected domain,  $J = \frac{A \cap B}{A \cup B}$ , where  $A \cap B$  is the size of the overlap and  $A \cup B$  the total length of sequence spanned by either the expected domain or the generated community.
- Perform an average of all the calculated Jaccard indices, weighted by the proportion of the total expected domain spanned by each module.

This gives a score for each domain in the protein, indicating how well it is reflected in the community structure. On simulated test data, this Jaccard performs sensibly (see Figure 4.1), giving high scores to close matches and low scores to poor matches. Note that like the original Jaccard, this score does not take values in the full range  $[0, 1]$  (see Section 2.4).

In order to calculate the significance of a given modified Jaccard, we use the z-score. This is defined as:

$$z = \frac{\tilde{J} - \mu}{\sigma}$$

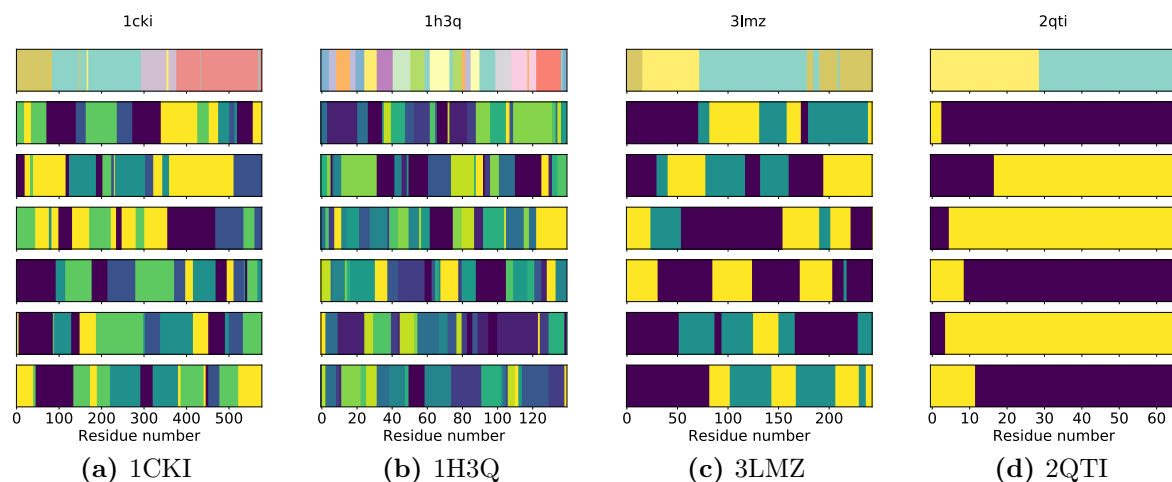
where  $\tilde{J}$  is the modified Jaccard index between the expected and generated partitions.  $\mu$  and  $\sigma$  are the average and standard deviation of the modified Jaccard between the expected partition and a set of null models.  $\mu$  therefore indicates the modified Jaccard expected by chance. A z-score of 2 indicates that the modified Jaccard between the



**Figure 4.1:** The performance of the modified Jaccard index on test data, with the expected domains above and example community structures below. Each expected domain is assigned a modified Jaccard,  $\tilde{J}$ . The score for the protein is then the average over all expected domains. We see a perfect score on perfect matching, with lower-quality matches scoring lower values. Note that the lower right figure, representing roughly the poorest imaginable case, still achieves a modified Jaccard of 0.2. This figure is taken from [131].

generated and expected partitions is 2 standard deviations higher than the expected value, and therefore corresponds to a p-value of  $\sim 0.02$  (assuming a normal distribution).

The null models used here are randomly generated, such that some key properties of the generated community structure are retained. In this work the null models are created by constraining the number of boundaries (changes from one community to another along the sequence), and the total number of communities. Boundaries and community labels are then placed randomly to obey these constraints. Figure 4.2 shows the community structure to be tested above, with 6 generated null models below. These models succeed in capturing the rough features of the generated structure, whilst preserving randomness.



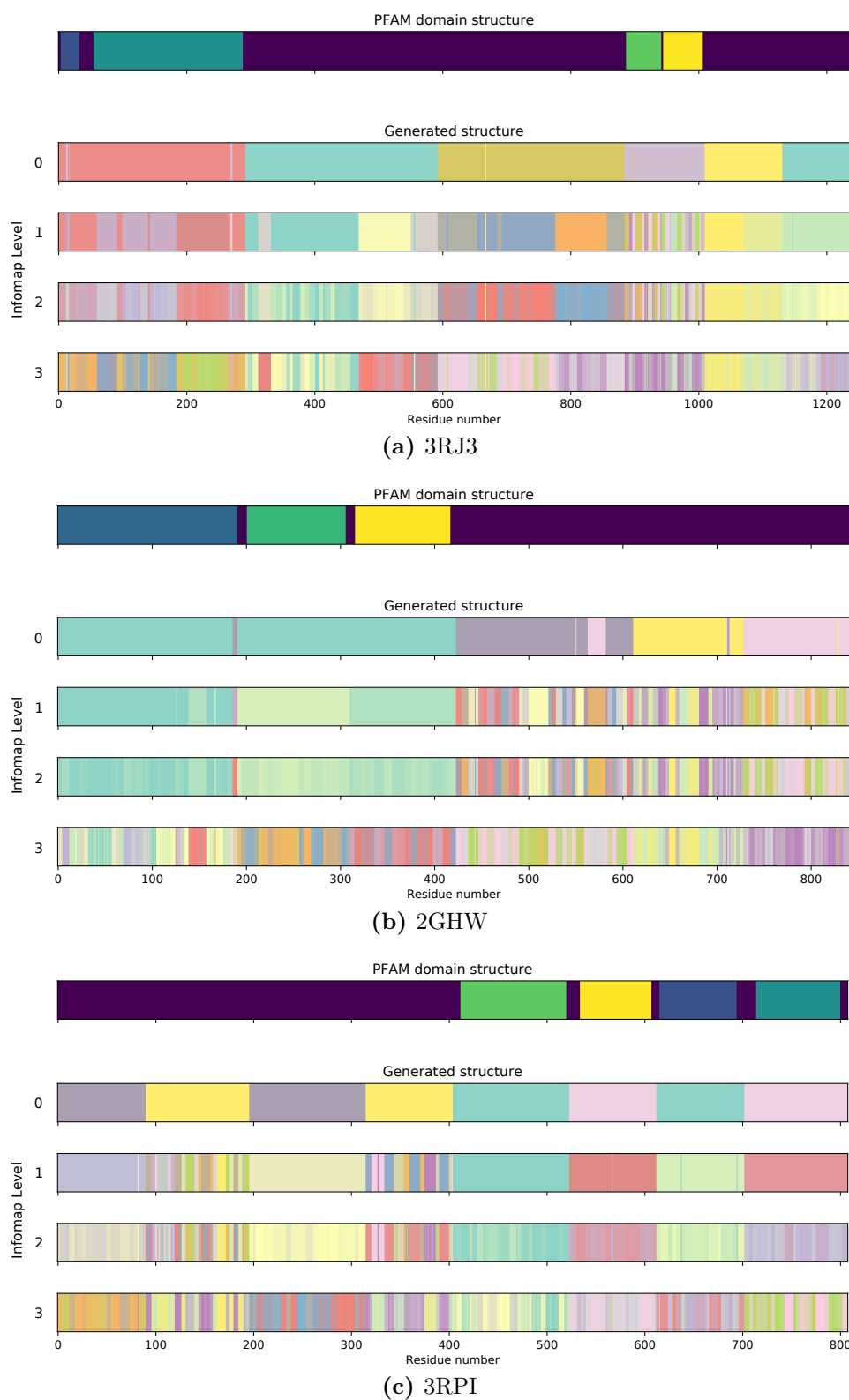
**Figure 4.2:** Randomly generated null models, such that the number of boundaries between communities, and the total number of communities, is preserved. The generated structure to be tested is shown above, with six null models shown below. These null models succeed in capturing the properties of the test structure, ensuring that the comparison between null model and test is fair. This figure is taken from [131].

## 4.2 Collating existing annotation schemes

In addition to a similarity metric, we also need to generate a dataset of existing annotation schemes, with the region of the PDB file that they correspond to. In order to create this resource, we use the PDBe REST API [132]. For a given PDB reference, this API allows you to request information on the regions of structure annotated by CATH, SCOP, Pfam, and others. By systematically querying this web resource and storing the information on each PDB studied, we incrementally generate a local dataset which can then be parsed and searched systematically.

## 4.3 Comparing communities to Pfam annotation

Plotting the Infomap-generated community structure and comparing to Pfam domains shows reasonable agreement for initial test proteins (see Figure 4.3).



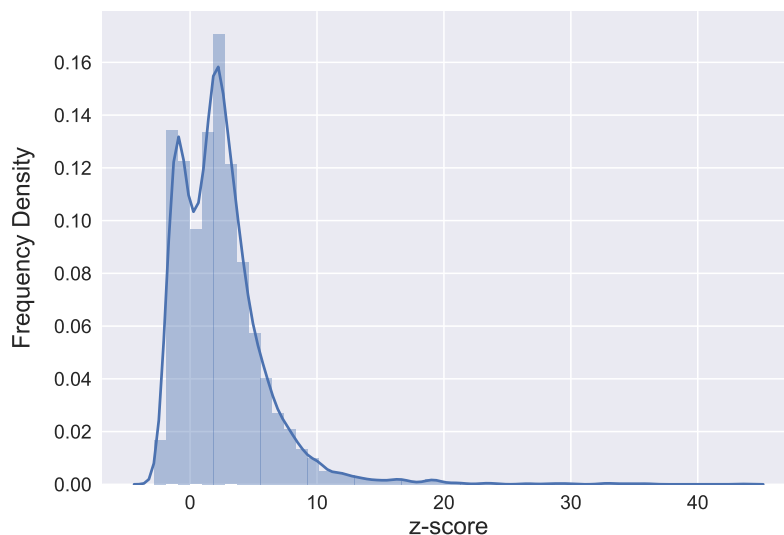
**Figure 4.3:** Comparison between Pfam domains and Infomap results for three test proteins, using a scaling factor of 4.5 on amino acid networks. All proteins show reasonable agreement between the Pfam domains and a level of the Infomap community structure, though note that the level giving the best match varies between proteins.

Extending this comparison to a larger set of  $\sim 1000$  proteins, Figure 4.4 shows that there is indeed a significant agreement in general between the Pfam domains and the generated community structure. We see a bimodal frequency density, with a reduced frequency density near  $z=0$ . The reason for this could not be established - no large-scale differences between the community or Pfam structures corresponding to  $z$ -scores near either modal value could be seen.

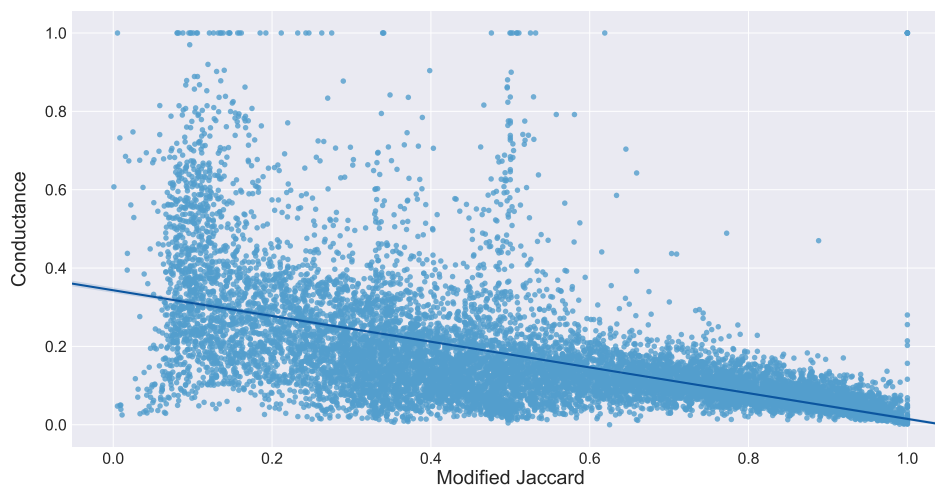
The agreement between Pfam domains and generated community structure is striking, as the communities found are based purely on the protein's structure (through its network representation), whilst the Pfam domains are based purely on sequence homology.

However, we expect discrepancies when the sequence domains correspond to more spatially extended regions of the structure, which are not well intra-connected. This can be measured by mapping the region of the protein corresponding to the Pfam domain onto the network, and calculating the conductance of that region. The lower the conductance, the more well-isolated an area of network the Pfam domain corresponds to (see Section 2.4). As such, we expect the Jaccard, which reports on the match between the Pfam domain and the generated community structure, and the conductance, to negatively correlate. Figure 4.5 shows this is indeed the case.





**Figure 4.4:** Histogram showing the distribution of z-scores, giving the significance of the match between the Pfam sequence domains and the community structure generated using Infomap. This figure is taken from [131].

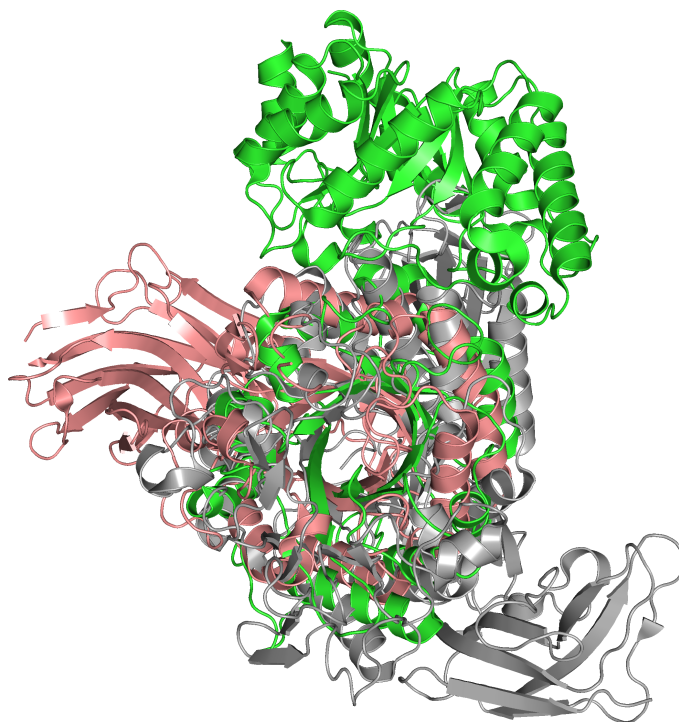


**Figure 4.5:** The conductance of the Pfam domain, when mapped onto the network, against the modified Jaccard (indicating how well it corresponds to the community structure). The conductance is 0 for perfectly-isolated communities, and 1 for communities fully connected to the rest of the network, so we expect a negative correlation between Jaccard and conductance; this is seen for the proteins studied here. This figure is taken from [131].

## 4.4 Comparing communities to SCOP annotation

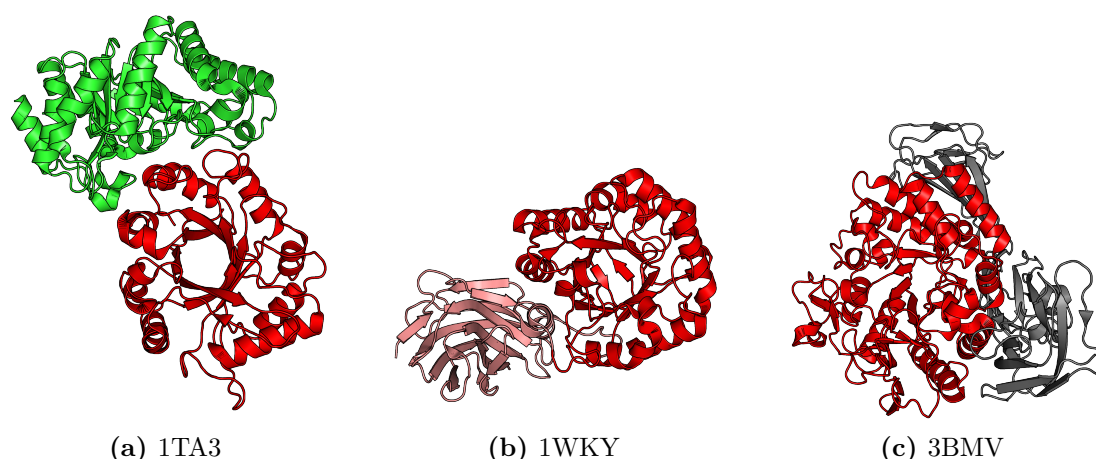
As the communities are created from purely structural information, a natural comparison is between the generated communities and the structure-based SCOP annotation set. In order to perform this comparison, we use a test set of proteins known to share a similar SCOP domain (superfamily c.1.8), but with distinct structure elsewhere.

The c.1.8 SCOP reference represents a superfamily of transferases - we expect these to share a catalytic core to carry out their function, but with distinct auxiliary domains (see Figure 4.6 and Figure 4.7).



**Figure 4.6:** Three members of the c.1.8 superfamily (PDB references 1TA3, 1WKY, 3BMV), aligned on their common SCOP domain. We see a common “ $\beta$ -barrel” structure, and similar arrangement of the helices within the aligned region. Outside the SCOP domain, the structures differ highly. The aligned regions have RMSD 8.286Å (between 1TA3 and 1WKY) and 5.743 Å (between 1TA3 and 3BMV).

The proteins are converted into networks of amino acids with scaling parameter  $s = 4$ , then community detection is performed with Infomap and the resulting communities are mapped onto the protein structure. We use the GLOSIM structural similarity algorithm (see Section 2.2) to compare the generated protein fragments to one another.



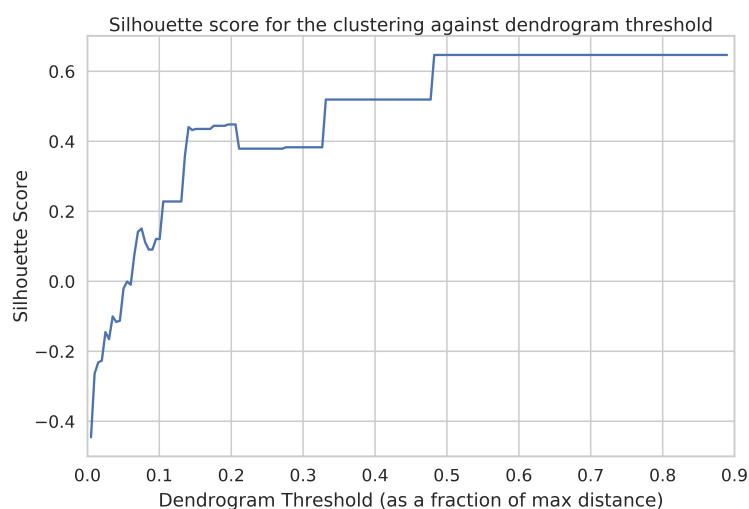
**Figure 4.7:** The same three members of the c.1.8 superfamily as in Figure 4.6, here showing how the SCOP domain relates to the full protein structure.

The GLOSIM similarity algorithm has been successfully used on crystal structures and small organic molecules, but has never been scaled to protein-sized structures. The GLOSIM algorithm uses the SOAP descriptor for each atom in the target structure, which specifies the atom’s local environment. The SOAP descriptors are then matched between pairs of atoms in the two structures to be compared, to generate an overall similarity metric. The SOAP descriptors have two key parameters:

- The cutoff radius,  $R$ . This determines which atoms are considered to be part of a target atom’s local environment. A larger cutoff radius will include more information, but has greater computational expense.
- The smoothing parameter,  $\sigma$ . For a target atom, we take all nearby atoms and replace them with Gaussians of width  $\sigma^2$ . This parameter therefore determines the sensitivity to small fluctuations. A smaller  $\sigma$  parameter requires a longer descriptor, as more spherical harmonic functions are needed to account for the variation in the density. As a result, decreasing  $\sigma$  will also increase computational complexity.

In the remainder of the chapter, we apply the GLOSIM algorithm to the alpha-carbons of each protein, and use  $R = 15\text{\AA}$  and  $\sigma = 1.5\text{\AA}$  (roughly the length of a carbon-carbon bond).

In order to find sets of fragments which are all structurally similar to each other, we use hierarchical clustering on the matrix of similarity scores (see Figure 4.9). Hierarchical clustering is an iterative process in which all protein fragments start in their own cluster, and clusters are merged according to their similarity. This results in a tree structure, or dendrogram, of cluster membership, which can be cut at any level to choose a set of clusters. We choose the clustering level based on the silhouette score [133], which



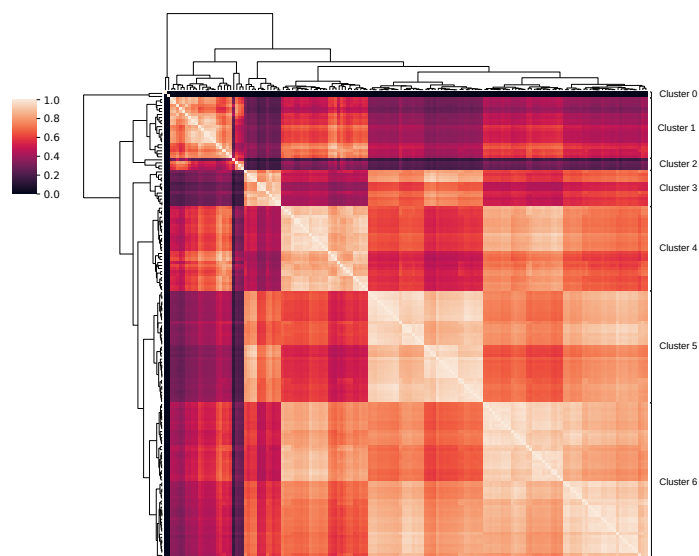
**Figure 4.8:** The silhouette score against dendrogram level for the 40 proteins in the c.1.8 test set. We see a clear ‘elbow’ point at a cutoff of 0.15 - this is the level at which we cut the dendrogram, giving the clusters shown in Figure 4.9.

maximises the difference between the intra-cluster variability and the inter-cluster distance (as shown in Figure 4.8). We choose this level by selecting the “elbow” point where the gradient sharply decreases. This method has been shown to be successful when combined with hierarchical clustering as a way to estimate the number of clusters in a data set [134]. Once this level is chosen (here 0.15 of the maximum dendrogram distance), we can extract the sets of similar protein fragments and compare to the known SCOP domains.

Figure 4.9 shows the seven sets of fragments at this dendrogram level, that are conserved at this across the SCOP superfamily.

We can investigate the statistics of these sets by plotting a histogram of module sizes for each cluster (Figure 4.10). Domains traditionally consist of 40-350 amino acids [24] - with the exception of clusters two and three, the fragments found here are within this range.

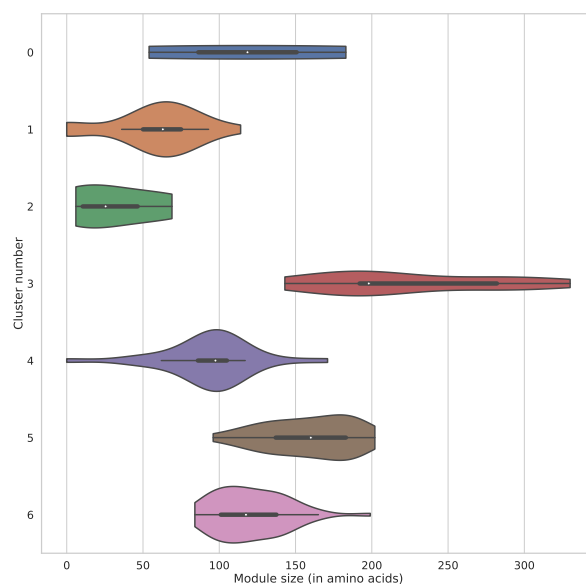
We can check the extent to which the modules are structurally similar by plotting a histogram of the root-mean-square distance (RMSD) between the aligned fragments within a cluster, and compare this to the RMSD of the chains that these fragments are taken from. In both cases, the alignment is performed using Pymol [128]. As shown in Figure 4.11, for each cluster we see a significant reduction in RMSD relative to the set of full protein chains.



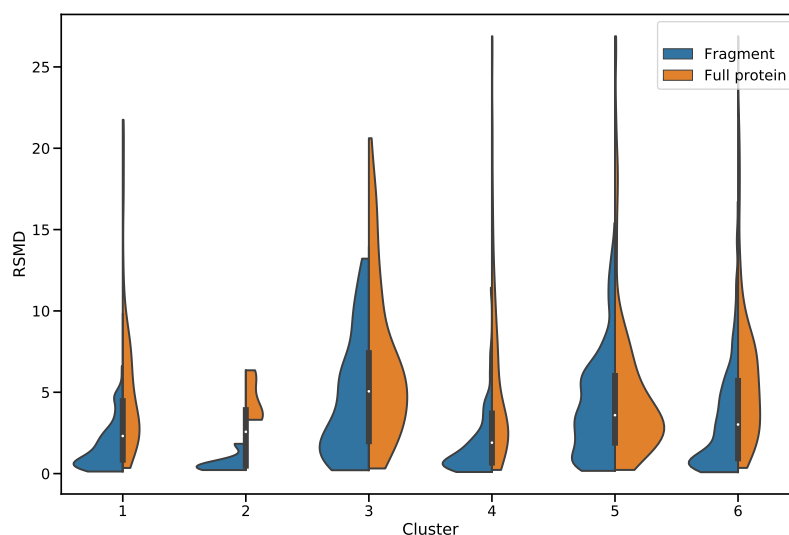
**Figure 4.9:** The clustered matrix of GLOSIM similarity scores for the protein fragments generated from a set of 40 proteins, all known to share a similar SCOP domain but with distinct structure outside this region. A lighter colour indicates a stronger similarity - we see there are clusters of fragments that are indeed highly similar to each other.

By structurally aligning the fragments corresponding to a given cluster, we can obtain a sense for how structurally similar the fragments are, and how they compare to the original c.1.8 domain. Here we look at cluster five (Figure 4.12). We see that the fragments in this cluster are extremely similar to each other. However, when we compare these generated fragments to the SCOP domain (Figure 4.13), we see that they correspond to a separate region of the protein structure.

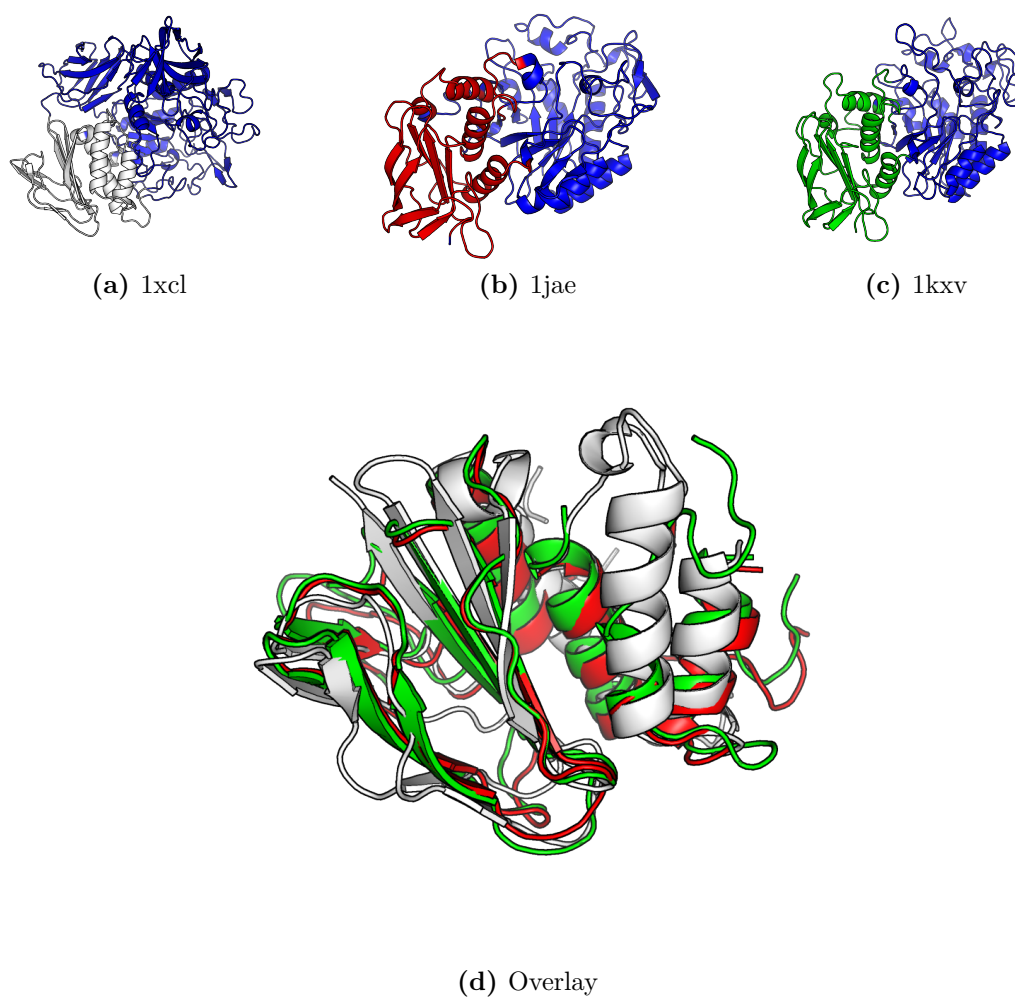
When we use our modified Jaccard index to compare these clusters to the SCOP, Pfam and CATH annotations to each structure, we see that none of these sets of conserved fragments individually match existing annotations, as shown in Figure 4.14.



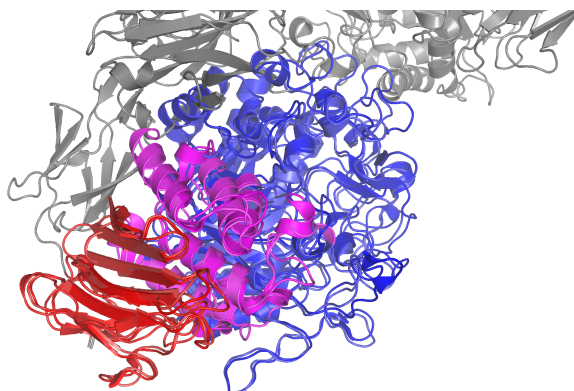
**Figure 4.10:** A violin plot showing the distribution of fragment sizes, measured in amino acids, for each of the seven clusters found. A domain is traditionally of size 40-350 amino acids [24] - the modules found here are broadly within this range.



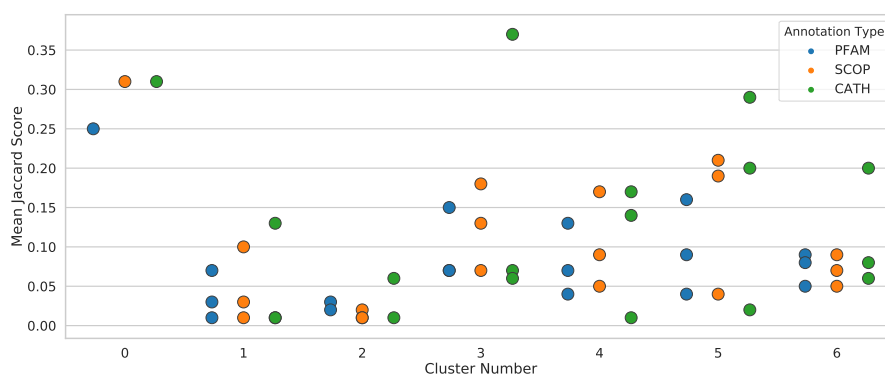
**Figure 4.11:** A violin plot showing the distribution of RMSD values for each pair of fragments in a cluster, for 6 of the 7 clusters (cluster 0 not shown). The left side of the distribution (blue) is the RMSD for the fragments, when aligned using Pymol. The right side of the distribution is the RMSD for the full protein chains. In all cases we see a reduction in the RMSD as a result of the fragmentation process.



**Figure 4.12:** Three fragments from the sixth cluster in Figure 4.9. Each proteins' structure is shown in blue, with the subregion isolated with community detection shown in white (1xcl), red (1jae) and green (1kxv). Below, we show these three subregions aligned, with an RMSD of 4.8 and 5.1Å.



**Figure 4.13:** The Infomap module and SCOP domains for the proteins 1CXL, 1JAE, and 1KXV as in Figure 4.12. The proteins are aligned on their Infomap modules. The Infomap module is given in red, the SCOP domain is blue, the region where they overlap in purple, and unannotated regions in gray. The Infomap module overlaps with the SCOP domain, but also includes a region unclassified by SCOP.



**Figure 4.14:** The mean modified Jaccard between the region of sequence corresponding to the generated fragment, and existing protein annotations (Pfam, CATH and SCOP), averaged over each community in the cluster of similar fragments, for the six clusters found in Figure 4.9. Only the Jaccard scores for the top three most similar annotations are displayed.

## 4.5 Communities as novel candidate domains

To explore the discrepancies between our modules and the structure-based SCOP domains, and attempt to find novel candidate domains which do not correspond to existing SCOP annotation, we extend our analysis to a larger subset of the PDB.

The ASTRAL compendium [135, 136] is a companion to SCOP which, in addition to other functionality, provides representative subsets of the PDB which span the set of



classified protein structures. Using this, we obtain a subset of the PDB in which the sequence similarity is less than 90 percent. This corresponds to 27330 proteins.

We then filter this dataset to a high-quality and high-resolution subset, using the AEROSPACI score (also a part of ASTRAL). The AEROSPACI score takes account the quality of the structural data used to generate PDB models; the resolution of the structure, the R-factor describing the model's fit to the experimentally determined electron density map, and the stereochemistry of the model. By selecting only those proteins with a score greater than 0.5, we limit ourselves to 4064 proteins.

Unfortunately, this process naturally biases our dataset towards proteins with fewer numbers of atoms, where we don't expect our method to generate sensible results. As such, we further filter for proteins known to have multiple domains in SCOP - a total of 422 single-chain proteins. This dataset is highly structurally diverse, with 433 unique SCOP classes represented in the dataset, 223 of these occurring only once.

When converted to networks with scaling parameter  $s = 4$ , and fragmented using Infomap, this results in 1573 fragments.

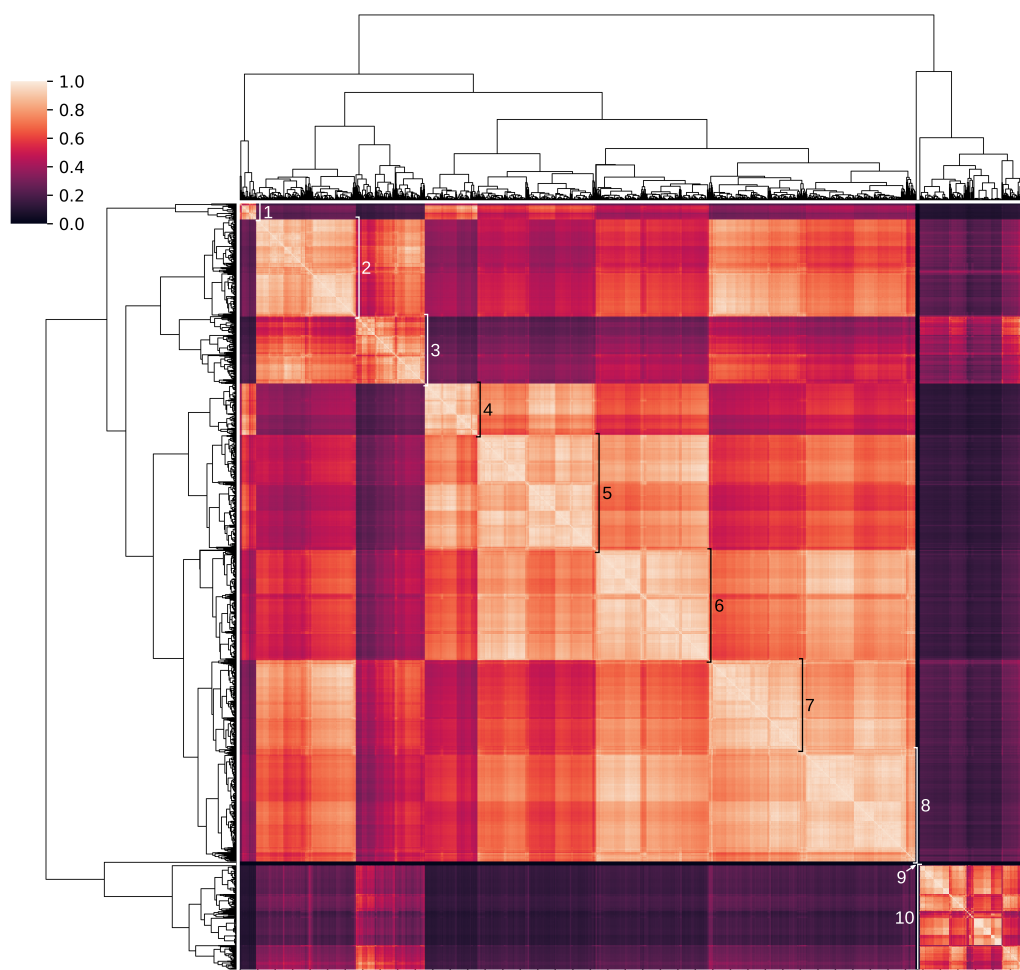
These can then be compared for similarity in an all-to-all fashion using GLOSIM, as previously, and the resulting score matrix clustered to identify sets of structurally similar fragments, as shown in Figure 4.15.

The dendrogram can be cut at any arbitrary point, and each will result in a different set of clusters. To obtain the optimal clustering level, as before, we plot the silhouette score [133] against the level of the dendrogram chosen, and choose the 'elbow' point.

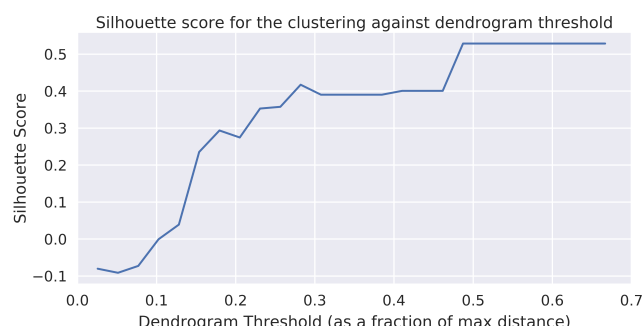
Choosing this point gives the dendrogram seen in Figure 4.17, resulting in 11 clusters of protein fragments, numbered 0 to 10.

We can investigate the properties of this set by comparison to existing classification schemes, as well as by aligning the fragments and visually inspecting. Plotting histograms of the module sizes (Figure 4.18) as before shows that the clustering process groups fragments into sets of broadly similar sizes - fragments within the same cluster tend to be within 25 residues of each other. We also see that the fragments are, with the exception of clusters 0 and 10, the correct size to be considered as domains.

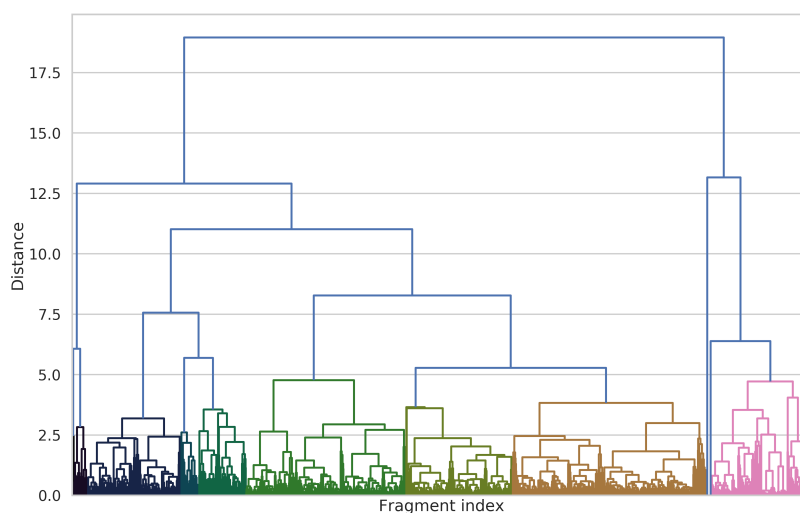
The Infomap-based community detection method does not reliably generate fragments with the desired properties on this extremely diverse dataset. We see that the generated fragments often don't correspond to compact, globular regions of the protein structure (Figure 4.20). We also find overpartitioning, i.e. that the protein's fragments are on



**Figure 4.15:** The matrix of similarity scores between the fragments found using Infomap on a test-set of 422 high-resolution protein chains, hierarchically clustered using the scikit-learn library [137]. The bracket indicates cluster membership, labelled by number (cluster 0 not included). A lighter shade indicates more similar fragments.

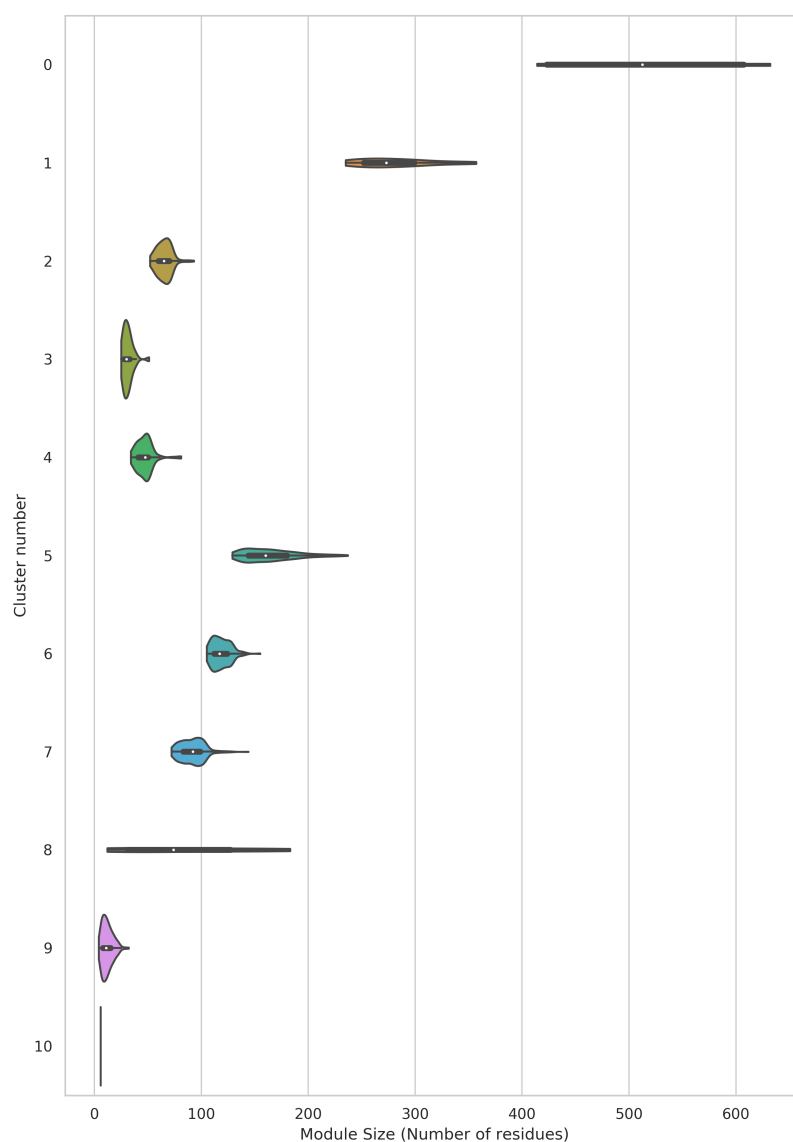


**Figure 4.16:** The silhouette score of the clusters against the level of the dendrogram chosen for the clustering. We see an ‘elbow’ at roughly 0.175.

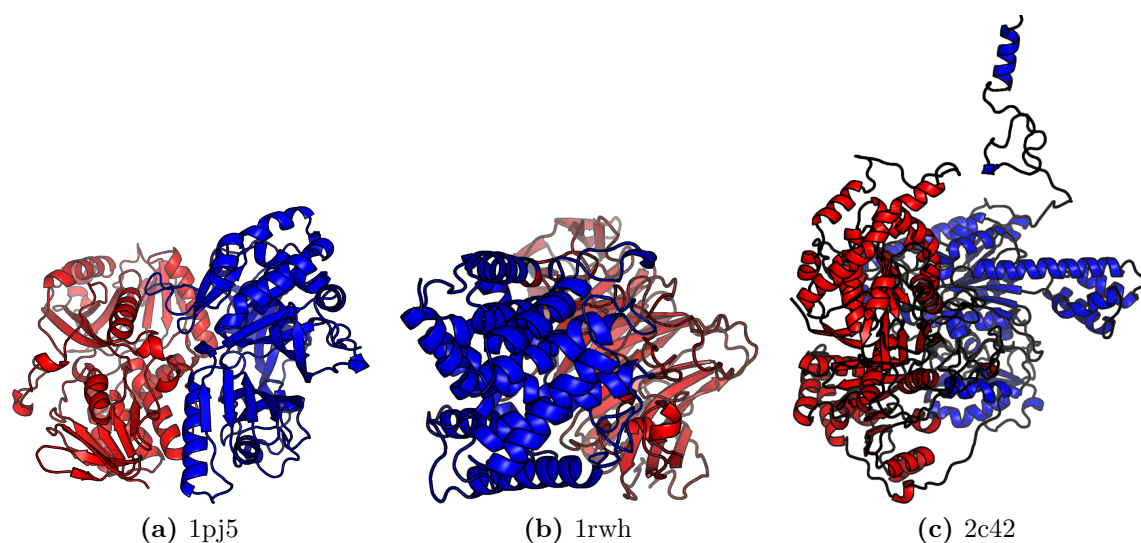


**Figure 4.17:** The clustering dendrogram, cut at a threshold of 0.175 of the maximum distance. Clusters are shown in distinct colours - a total of 11 are seen here.

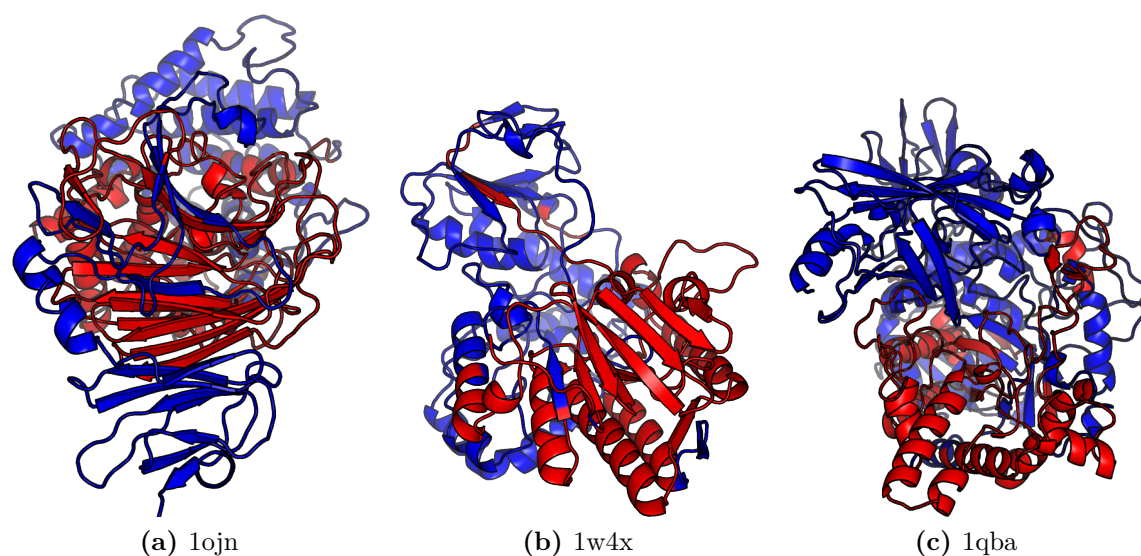
a length scale too small to be useful. In addition to these issues with the community detection process, the GLOSIM structural similarity scoring appears to give surprisingly high scores to intuitively mismatching fragments (such as in Figure 4.21, where all-beta and all-alpha fragments are grouped). Example fragments for the full set of protein clusters are given in Appendix A.4.



**Figure 4.18:** Violin plots showing the distribution of the fragment sizes, in number of residues/amino acids, for each of the 11 clusters found. We see that the fragments within each cluster have broadly similar sizes, and we also see, with the exception of clusters 0 and 10, that the fragments are approximately the right size (typically domains tend to range from 40-350 amino acids in size).



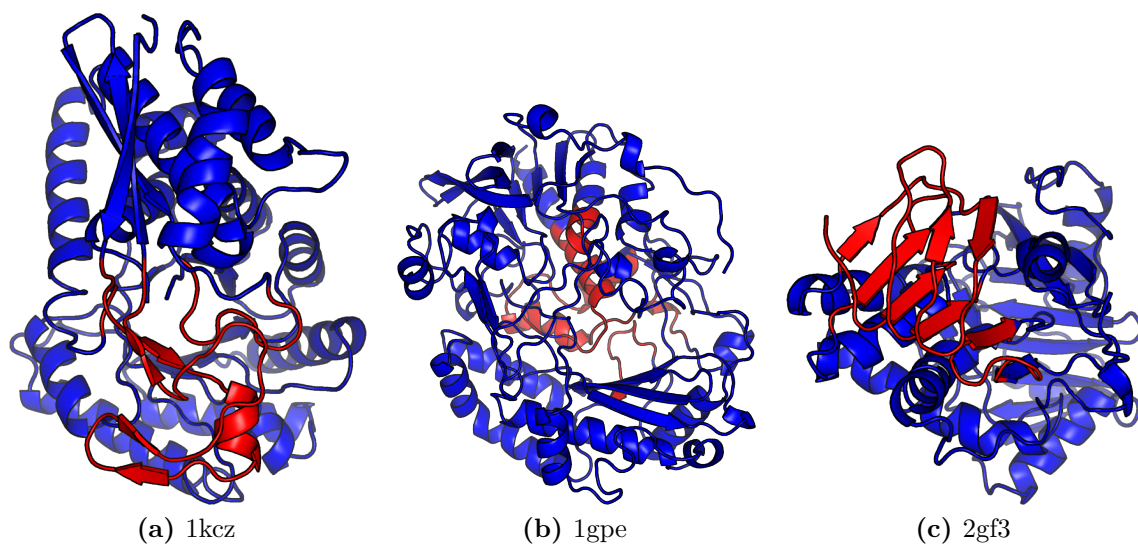
**Figure 4.19:** Selected protein fragments from the multi-domain test set, in cluster 0, demonstrating underpartitioning.



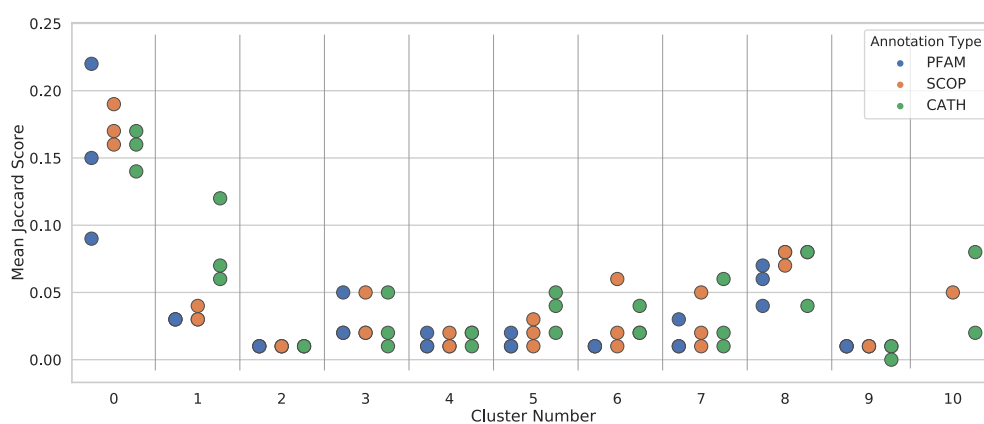
**Figure 4.20:** Selected protein fragments from the multi-domain test set, in cluster 1. We see correctly sized fragments, but the fragments are not contiguous regions of the protein's structure.

In agreement with manual inspection of the fragments, the overlap between the fragments generated and the known SCOP, CATH and Pfam annotations using the modified Jaccard is extremely low (see Figure 4.22).

Overall, we see that the communities generated behave as expected when compared to the Pfam sequence domains, and generate promising results on the c.1.8 SCOP superfamily test set, correctly identifying structurally similar subregions. The method



**Figure 4.21:** Selected protein fragments from the multi-domain test set, in cluster 2, demonstrating overpartitioning, and showing that dissimilar fragments may still give high GLOSIM scores (notice 2gf3 is an all-beta fragment whilst 1gpe is all-alpha).



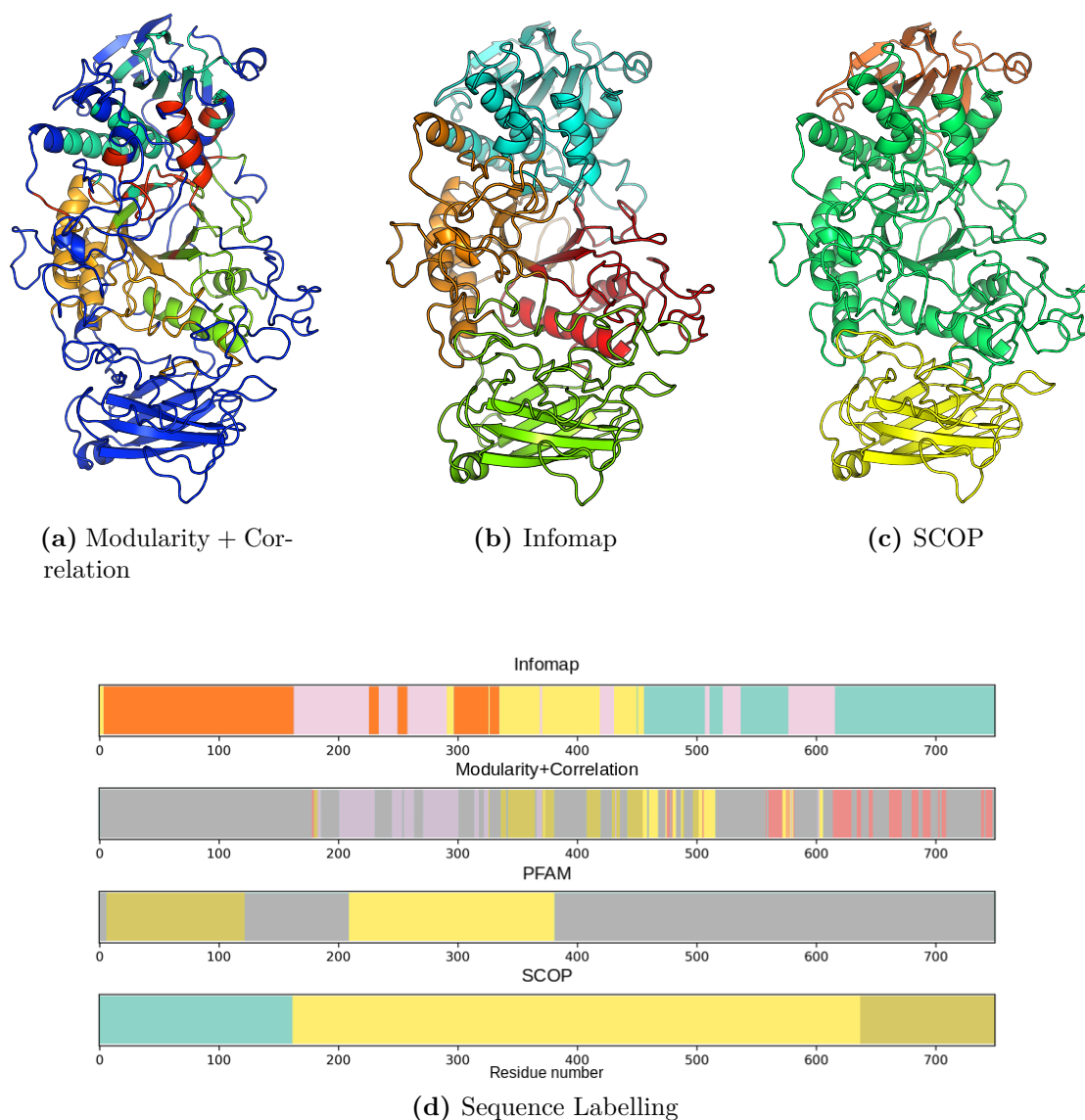
**Figure 4.22:** The overlap between the areas of sequence corresponding to the generated fragments, and the existing annotations (SCOP, CATH, and Pfam), as given by the modified Jaccard score. As before, only the scores for the top three most similar annotations of each type are given.

breaks down on the extremely structurally diverse dataset - however, this is equally likely to be a failure of the hierarchical clustering method as the community detection process, as the number of unique SCOP domains in the dataset suggest that there will be few structurally repeated subregions in the dataset.

## 4.6 Comparison to existing methods

Following the assessment of the Infomap-based method in its own right, here we compare its performance to previous work, based on correlation networks [119]. This method uses a set of homologous proteins to calculate the correlation between the amino acid positions in the set - this results in a single network, with the correlation defining the edge weights. A drawback of the correlation-based approach is that a set of homologous proteins is needed; our method has the advantage that it can be performed on single proteins, meaning that the partition spans the full protein structure, making the approach scalable to larger datasets.

However, even when the two methods are directly compared, the Infomap-based approach yields partitions with a closer resemblance to the SCOP annotations. Figure 4.23 compares these results qualitatively on a single reference protein, to SCOP annotations and to the results obtained using our protocol. We see that the correlation-based approach leaves large sections of the protein unannotated - in addition, the method fails to discover auxiliary variable regions. Figure 4.24 compares the results over the full set of proteins described in [119], using the z-score. We see a more significant correspondence for both Pfam and SCOP using the Infomap-based approach.’

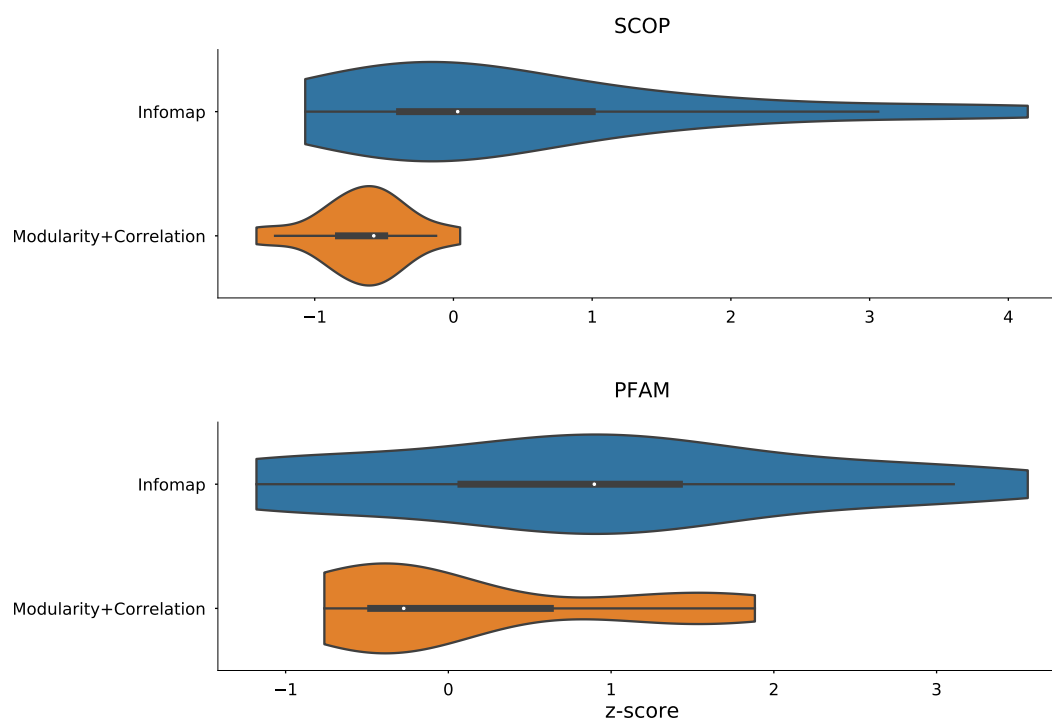


**Figure 4.23:** A comparison of annotations of the protein 1BF2. (a) The decomposition generated in previous work using a modularity-based method, combined with residue correlation analysis [119]. Grey regions correspond to un-annotated regions of the structure. (b) The decomposition using Infomap presented in this chapter. (c) The domains listed in the SCOP structural domain database. (d) The same comparison, along with the Pfam annotations, presented as labellings of the protein sequence. Again, dark blue represents unannotated regions of the sequence. This figure is adapted from [131].

## 4.7 Community structure at a sub-domain length scale

Given the relative success of partitioning protein structure at the level of domains, here we explore possible uses for communities at smaller length scales. We have previously





**Figure 4.24:** A comparison of the z-scores for a set of twenty reference proteins, giving the significance of the overlap between SCOP (above) and Pfam (below) for the modularity+correlation method [119] and the proposed Infomap-based method. We see that in both cases the Infomap-based method has a more significant similarity to existing annotations. This figure is taken from [131].

applied community structure at this smaller scale to identify parsimonious descriptions of inorganic crystal structures [138], using the AFG method (see Section 2.4).

One aspect of protein structure that makes modelling difficult is the extremely large configuration space occupied by the protein. This space can be reduced by using a description in terms of the relative orientation of protein fragments. The extent to which this compression occurs can be quantified by calculating the effective degrees of freedom (DoF) of the protein.

The maximum number of degrees of freedom of an  $N$ -atom structure,  $\mathcal{I}_{max}$ , is  $3N - 6$ . This corresponds to each atom moving freely in three dimensions, with global rotation and translation constrained. However, we can reduce the effective DoF by using a modular structure, in which the protein can be specified by the relative position and orientation of its modules, plus the positioning of each atom within each type of module. A reduction occurs whenever there are repeated modules.

If a structure is divided into  $M$  modules, of which  $M_1$  have a single atom and  $M_2$  two atoms, the total DoF in the position and rotation of the modules is  $6M - 6 - M_2 - 3M_1$ , as each module with more than three atoms has 6 DoF. Each module with two atoms has 5 DoF (3 positional, 2 rotational) and each module with one atom has 3 DoF (positional only).

The DoF required to describe each module’s internal structure is  $3N_i - 6$ , where  $N_i$  is the number of inequivalent atoms in module  $i$ , if the module has more than three atoms. Two-atom modules have one internal DoF, the absolute distance between the two atoms. One-atom modules have zero internal DoF. The total DoF required to describe the system, in terms of its modules, is then the sum of the positional/rotational DoF of the modules and the internal DoF of each unique module:

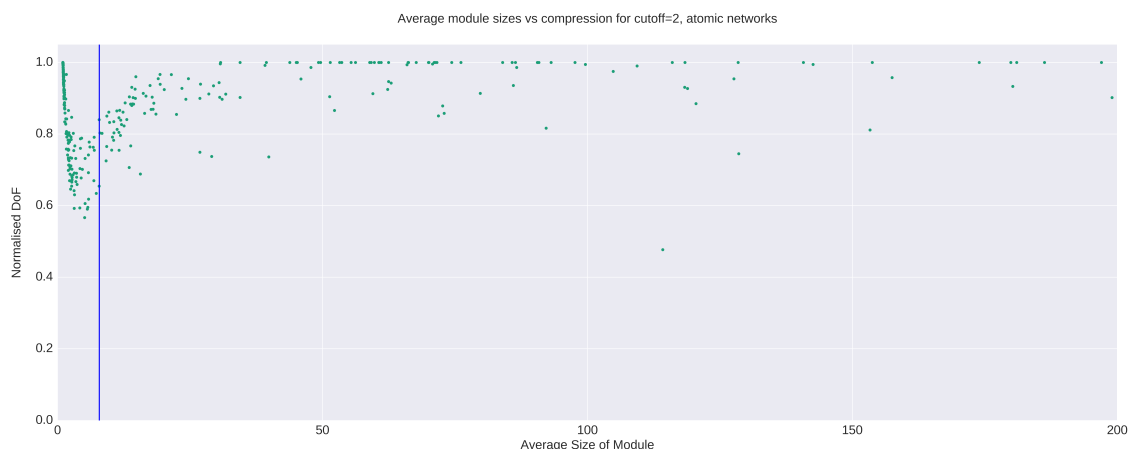
$$\mathcal{I} = 6M - M_2 - 3M_1 - 6 + \sum_{i=1}^{M'} \left( 3N_i - 6 + 3\delta_{1,N_i} + \delta_{2,N_i} \right)$$

where  $M'$  is the number of unique modular structures. We can establish which modules are unique by generating a SMILES string [139] using OpenBabel. This encodes the chemical structure as a unique text string. We assume that two modules are copies if they share a SMILES string.

Normalising this by  $\mathcal{I}_{max}$  gives a result  $\in [0, 1]$  giving the compression achieved by using a modular description. Since compression of information occurs when modules repeat, we expect a description in terms of amino acid residues to result in high compression; other regions of high compression may reveal relevant structural motifs within the protein.

Calculating the decomposition obtained when using amino acids results in a normalised DoF ( $\mathcal{I}_N$ ) of  $\sim 0.3$ . This may suggest possible information-theoretic grounds for the twenty-amino-acid alphabet used in protein structure.

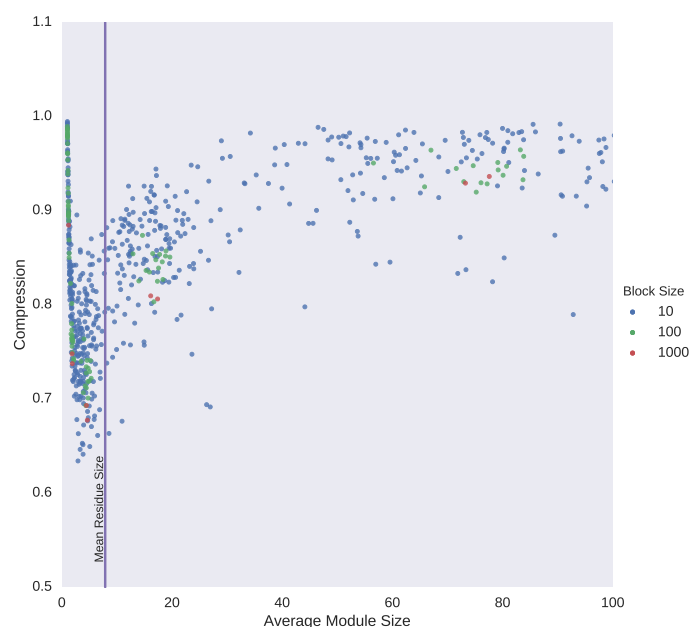
For a test set of proteins, we can plot the compression achieved for a given partition against the average community size in that partition. We see that  $\mathcal{I}_N$  is most often 1 (atomic networks). This is due to the lack of repeated communities within a single protein. However, a clear reduction in  $\mathcal{I}_N$  is seen when the communities are roughly the same size as amino acids. This also suggests that the most compact description of a protein is found by using amino acids as the building blocks.



**Figure 4.25:** The normalised degrees of freedom  $\mathcal{I}_N$  for a protein using a modular description, against the average module size for that modular description. The average size of an amino acid in atoms is shown as a blue line. A clear dip in  $\mathcal{I}_N$  is seen as the module size becomes comparable to the amino acid size.

The same compression can be calculated across sets of proteins. As the number of proteins compressed increases, we expect the number of repeated modules to increase, and with it the compression. Figure 4.26 shows that this is the case - however, as before,  $\mathcal{I}_N$  remains close to 1.

Overall, we see that coarse-graining using the smaller communities does result in a reduced configuration space for the protein, with the effective degrees of freedom reduced by 40%. However, this remains less effective than coarse-graining using the residues. This is perhaps unsurprising - the Infomap communities are constructed to optimise a different quantity, which may only weakly correlate with the objective measured here.



**Figure 4.26:** The normalised degrees of freedom  $\mathcal{I}_N$  for a protein using a modular description, against the average module size for that modular description, across sets of proteins of different sizes (10, 100, and 1000 proteins). Increasing the set size increases the number of repeated modules found, thus increasing the compression, but the trend remains as in Figure 4.25.

## Chapter 5

# Community structure as a proxy for topology

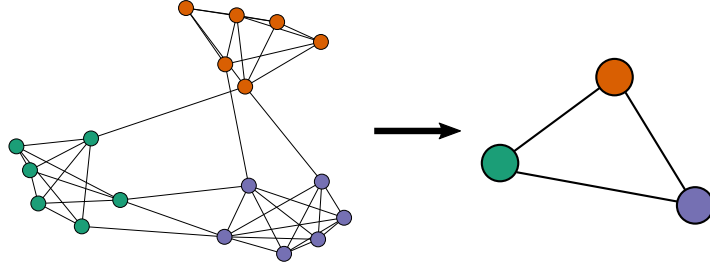
We have explored the communities found using Infomap as candidate domains; in this chapter, we turn to analysing the connections between domain-level fragments to reveal common global patterns underlying multi-domain protein architecture. We show that these global patterns are associated with GO descriptors of protein function.

### 5.1 Generating and describing protein super-networks

The arrangement of the protein's modules can be quantified by creating a network in which the protein's communities become nodes, linked if the respective communities are neighbours (known in the literature as a condensation network, coarse-grained network or super-network), as shown in Figure 5.1. By checking whether the relevant super-networks are isomorphic (if the nodes of one network can be relabelled such that the edges of the second network are recovered), we can group the set of proteins into classes which share the same super-network. Here we use the isomorphism checks built into the NetworkX package [140].

Once the super-networks are generated, and the separation into classes performed, GO term enrichment is used to investigate whether the groupings are functionally relevant (see Section 2.1).

From an initial set of proteins, we construct their supernetworks from the top level of the Infomap community structure. From there, we identify a subset which have isomorphic supernetworks. Each protein is associated with a given number of GO terms. Each



**Figure 5.1:** Example of the supernetwork generation process. Given a network with three communities (coloured on the left), we create a three-node supernetwork where the nodes are linked if there are edges between their corresponding communities.

GO term will occur a certain number of times in the wider dataset. So for a given GO term, we can work out the likelihood that the given number would appear by chance in the subset, and thus obtain a p-value (the following analysis adapted from [131]). This is done by taking the cumulative distribution function of the hypergeometric function (which is equivalent to the binomial distribution, but sampling without replacement). Let  $n$  be the number of proteins in the subset, and  $N$  the total number of proteins. For a given GO term, let  $k$  be the number of times it occurs in the subset, and  $K$  be the number of times it occurs across the full dataset. Then the likelihood that the term would be seen  $k$  times by chance is:

$$CDF = 1 - \frac{\binom{n}{k+1} \binom{N-n}{K-(k+1)}}{\binom{N}{K}} {}_3F_2 \left[ \begin{matrix} 1, & k+1-K, & k+1-n \\ k+2, & N+k+2-K-n \end{matrix} ; 1 \right]$$

where  ${}_3F_2$  is the generalised hypergeometric function. We conduct the test for each GO term and as such, need to adjust for multiple testing. We choose to apply the Bonferroni correction. If comparisons of  $M$  GO terms are being made, the raw p-value must be multiplied by  $M$  to give a more conservative estimate of the likelihood.

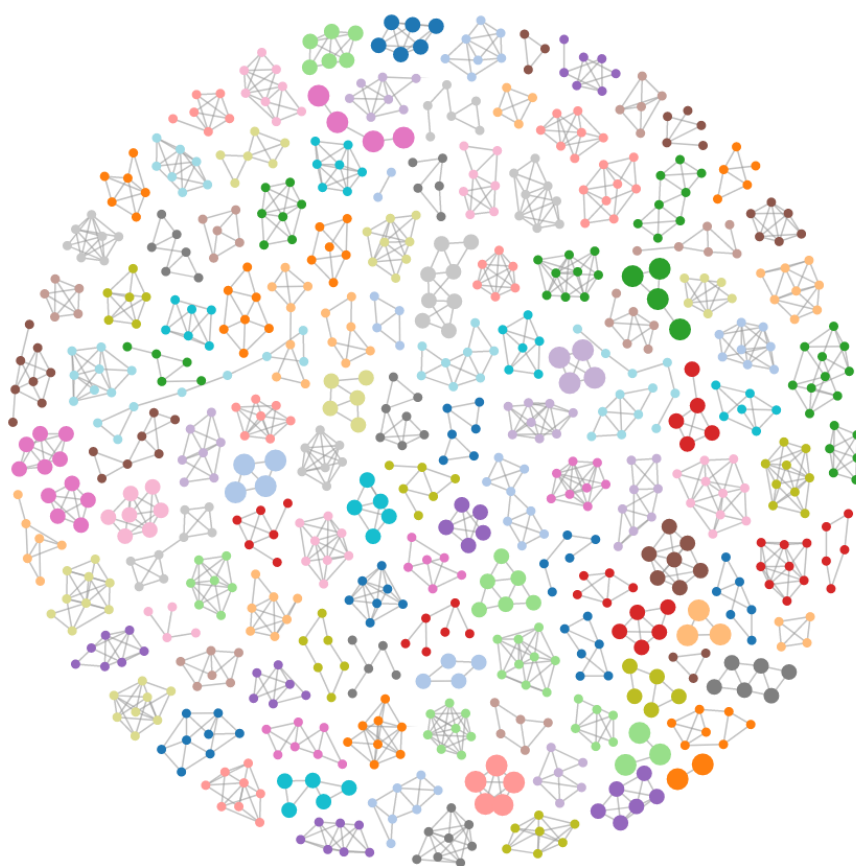
Selecting only the terms with a corrected p-value less than 0.01 gives a list of GO terms that appear significantly more often within the subset than would be expected by chance.

## 5.2 The properties of protein super-networks

When we perform the supernetwork generation process on a set of  $\sim 12,000$  multi-chain proteins, we see the topologies shown in Figure 5.2. This classification is displayed using Javascript and D3.js [141] at [wpg.io](http://wpg.io). Double-clicking a super-network reveals all the pro-

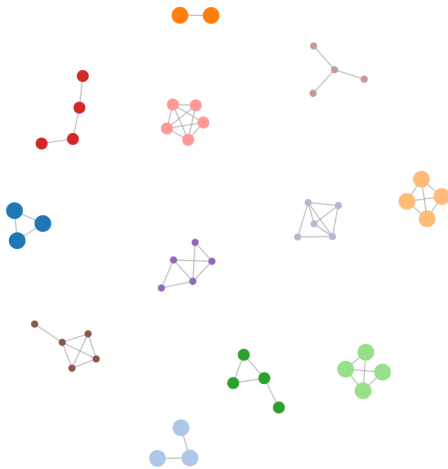
teins belonging to that super-network, listed by PDB reference, along with the enriched GO-terms. For example, the top five unique GO terms associated with the five-node clique are `GO:0008872` (glucarate dehydratase activity), `GO:0009098` (leucine biosynthetic process), `GO:0003862` (3-isopropylmalate dehydrogenase activity), `GO:0004109` (coproporphyrinogen oxidase activity) and `GO:0006725` (cellular aromatic compound metabolic process)

We see a large number of terms with  $p < 0.01$  for each supernetwork.













**Figure 5.2:** The set of super-networks resulting from community detection on a set of roughly 12,000 multi-chain proteins. We see a wide array of protein shapes and sizes. This visualisation is interactive at [wpg.io](http://wpg.io). Double-clicking a network lists the proteins, including the enriched GO terms found in that class. The node size indicates the number of proteins with that supernetwork.

When we consider each protein chain individually (as in Chapter 4), we see that 90% of the protein chains can be represented using only 10 super-networks, all of which are associated with GO-term enrichment (see Figure 5.3 and Table 5.1).



**Figure 5.3:** The super-networks generated from the community structures of  $\sim 4300$  protein chains, taken randomly from a non-redundant subset of the Protein Data Bank. Only super-networks common to at least 3 proteins are shown (accounting for  $\sim 3900$  proteins in total). The node size is proportional to the number of proteins exhibiting that super-network. This figure is taken from [131].

Super-network	Number of enriched GO Terms (p<0.01)	Number of proteins
	331	1725
	130	307
	116	841
	104	445
	34	55
	52	207
	48	26
	23	19
	9	6
	22	8

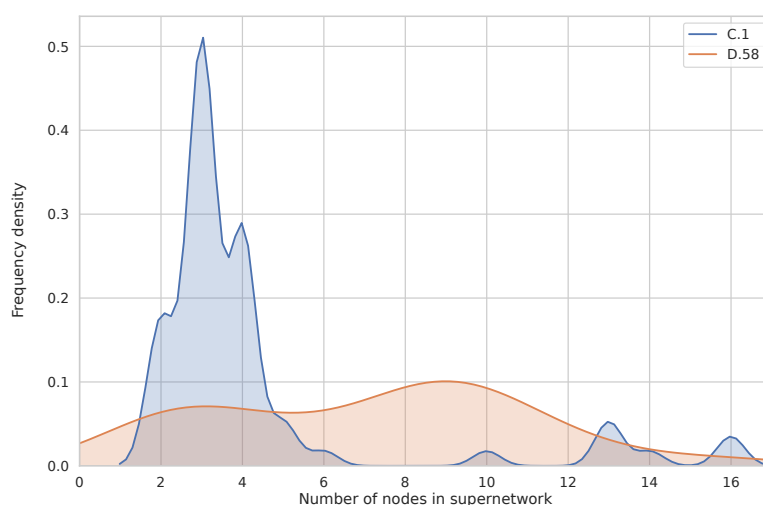
**Table 5.1:** The ten most common protein topologies in the data set studied in Figure 5.3, ordered by prevalence. This table is taken from [131].



### 5.3 Network vs structure similarity

GO-term enrichment suggests that the grouping according to supernetwork is capturing functionally relevant information. As it is likely that proteins sharing structural domains will perform similar function, we predict that SCOP class and supernetwork topology will be co-variant.

Initial tests suggest that members of different SCOP classes have distinct supernetworks: from the high-resolution dataset of proteins with less than 90% similarity introduced in Chapter 4, we select 100 members of the *c.1* and *d.58* Folds (chosen because these Folds are well represented in the dataset). Class *c* and *d* correspond to proteins containing both alpha-helices and beta-sheets, with parallel and anti-parallel beta sheets respectively. Comparing the number of communities obtained, we see a clear difference between the two classes (see Figure 5.4), as well as a large degree of heterogeneity for each Fold.



**Figure 5.4:** A histogram of the number of nodes in the supernetworks formed by members of fold *c.1* vs fold *d.58* - this is equivalent to displaying the number of communities.

### Implementing the maximum common subgraph similarity measure

In order to further compare network similarity and structural similarity, we implement a network similarity measure based on the Maximal Common Subgraph (MCS). This

has been shown to behave as a distance metric between graphs [142], and is valid for the unlabelled networks used here.

The similarity between two graphs  $G$  and  $H$  is defined as:

$$s(G, H) := \frac{|\text{MCS}(G, H)|}{\max(|G|, |H|)}$$

where  $|G|$  is the number of nodes in the graph  $G$ , and the MCS is the largest subgraph found in both networks. Finding the MCS is known to be a NP-complete problem. The algorithm implemented here is the simplest possible; starting from all  $N$  nodes of the smaller graph, say  $G$ , check for isomorphism with all  $N$ -node subgraphs of  $H$ . If no isomorphism is found, generate all subgraphs formed from  $N - 1$  of the nodes, and repeat.

This algorithm has worst-case time complexity:

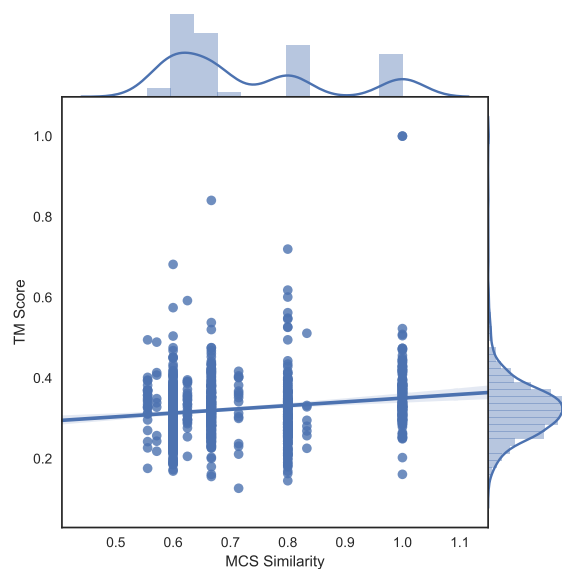
$$\sum_{i=1}^N i \binom{N}{i} \binom{M}{i} f(N, M)$$

where  $N$  is the size of graph  $G$ , and  $M$  the size of graph  $H$ , where  $N < M$ , and  $f(N, M)$  is the time complexity of the isomorphism check. In this case, the VF2 algorithm is used [143]. This has worst-case time complexity  $\mathcal{O}(M! \cdot M)$ . Graphs with under 30 nodes can be compared using this MCS algorithm in a reasonable time.

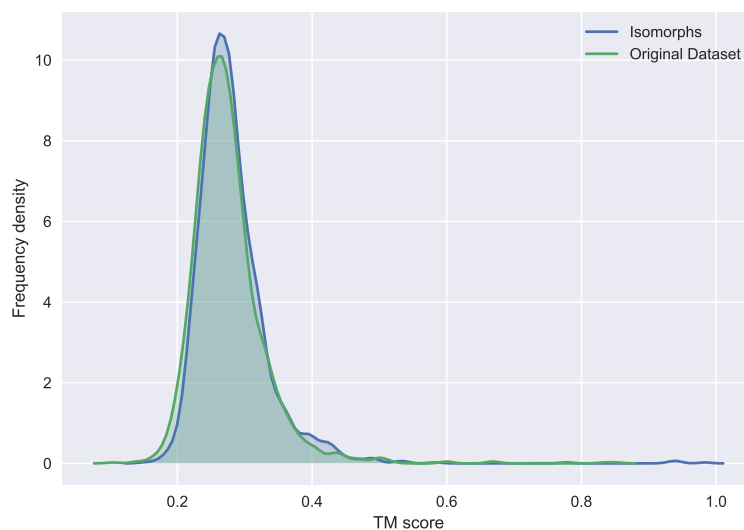
If the grouping by supernetwork is indeed into classes with the same broad topology, then we expect the MCS similarity to correlate well with the structural similarity, as measured by the TM-score [45], which uses a weighted version of the root-mean-square distance (RMSD) in order to more accurately compare global topology. However, as shown in Figure 5.5, there is only very weak correlation between the two measures.

We would also expect proteins which share a given super-network to show a greater structural similarity on average than proteins in distinct classes. Figure 5.6 shows only very slight evidence that this is the case.

Taken together, Figure 5.6 and Figure 5.5 suggest that whilst the grouping according to super-network is functionally relevant, the proteins' structures differ highly. This could be related to the breadth of the classes, or to the crude nature of the MCS measure.



**Figure 5.5:** The super-network similarity (as measured by the Maximal Common Subgraph) against the structure similarity (as measured by the TM-score). A very weak correlation is seen. The discretisation along the x-axis is due to the fact that the networks under study are small, such that the MCS is a simple fraction.



**Figure 5.6:** The structural similarity (as measured by the TM-score) between proteins with isomorphic super-networks (blue) and between the original dataset (green), showing only a slight increase in similarity for proteins with isomorphic supernetworks.

Alternatively, the issue could lie in the fact that proteins with highly differing sizes could have the same network structure at different levels

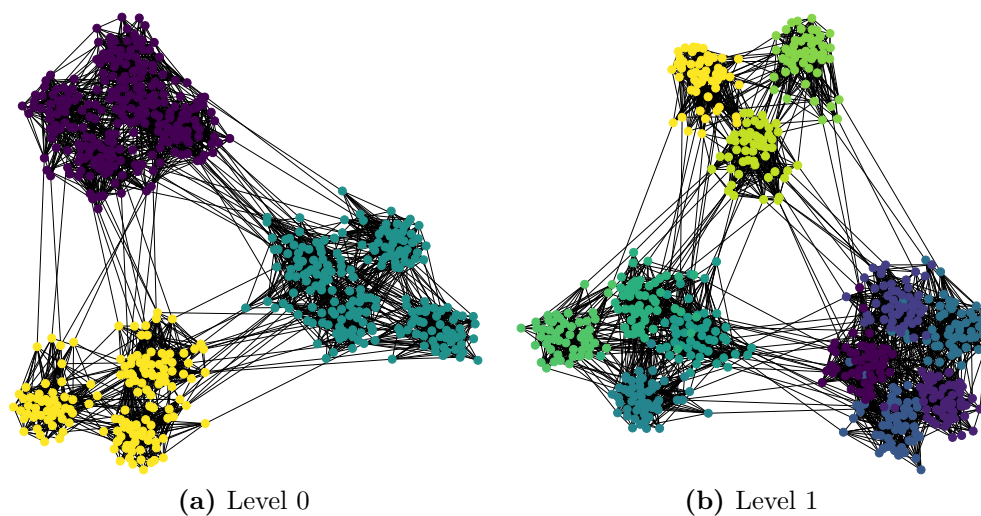
To address this issue and enable comparison of networks across several different levels, a measure which attempts to provide a better comparison is introduced in the remainder of the chapter.

## 5.4 Iterative subtree similarity

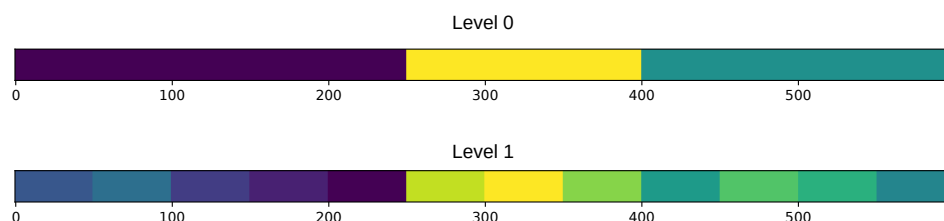
The previous comparison between the community structures of different proteins used only a single level of the proteins' community structure (in practice, the level with the largest communities). However, the community structure returned by Infomap is hierarchical.

We can extend the notion of the super-network given previously to the full hierarchical community structure. From there, we implement a novel comparison method based on iterative subtree labelling. The method proposed here is applicable to any instance where two hierarchical community structures are compared.

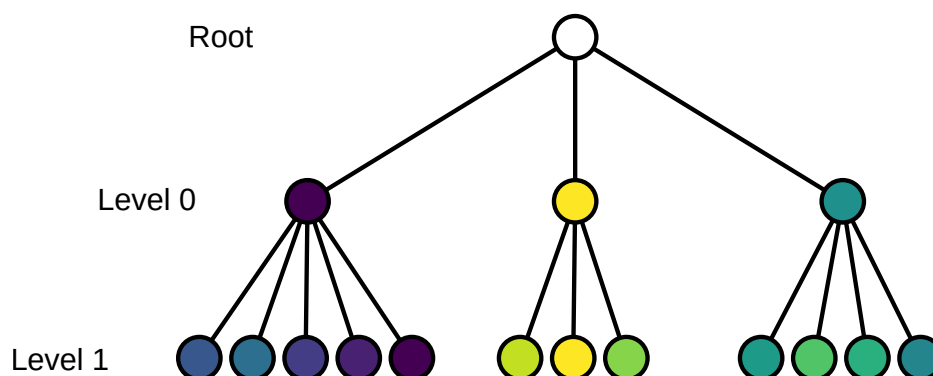
For single-level community structure, the super-network is generated as in Figure 5.1. For multi-level community structure, as shown in Figure 5.7 and Figure 5.8, we proceed as follows. From the hierarchical community structure, we generate a tree in which each community is a node. Communities are linked, between hierarchy levels, if one community is a sub-community of the other (see Figure 5.9). This is akin to the dendrograms seen in traditional hierarchical clustering methods.



**Figure 5.7:** Example of a network with multi-scale hierarchical community structure. Each of the three broad communities (in level 0) is itself made of a variable number of smaller communities (in level 1). Running Infomap on this network will result in a set of partitions for each level of the community structure (see Figure 5.8).

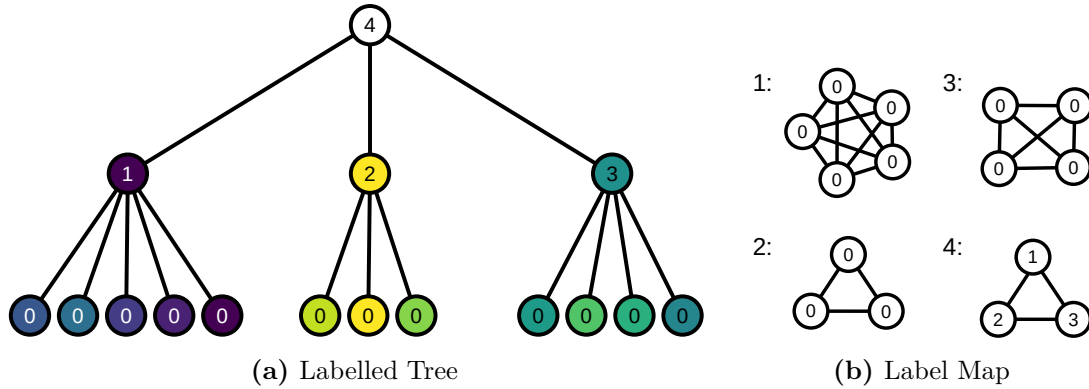


**Figure 5.8:** The output of Infomap when applied to the network shown in Figure 5.7 - two vectors of community membership. Distinct colours indicate the different communities.



**Figure 5.9:** The tree structure generated from Figure 5.8 - From the root node (indicating the single-community partition), we add a tree level for each level of the community structure. Nodes are coloured according to the colour of their original communities.

Once this tree has been created, we perform a bottom-up labelling process. All the “leaf” nodes (those within sub-communities) receive the label 0. We then proceed one level up the dendrogram, and label each node according to the super-network formed by its children, maintaining a map of which label each unique supernetwork corresponds to. For instance, the first node (in purple) on the first level of the community structure has five sub-communities. In the original network, these sub-communities are all connected to each other in a clique. We add an entry in our map (e.g. label "1") for this five-member clique of nodes with label 0, and proceed. If, when labelling a different node, another five-member clique of nodes with label 0 is seen, this is given the same label as previously. In this way, we build a labelled tree, along with an index showing which supernetwork each label corresponds to. Figure 5.10 shows an example labelling.



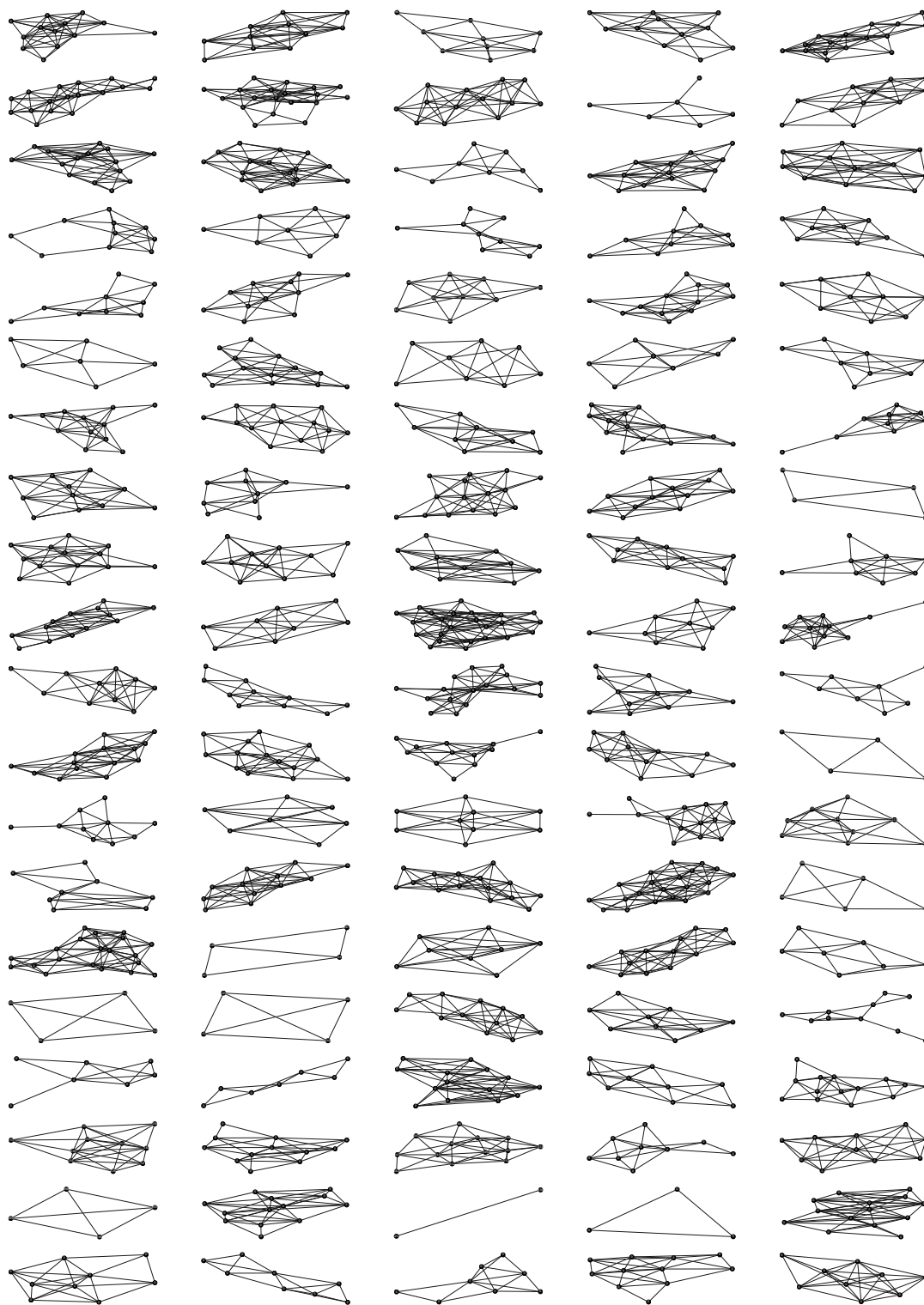
**Figure 5.10:** The result of bottom-up labelling of the tree in Figure 5.9. Leaf nodes are assigned a label 0, then each node above is assign a label according to the relative arrangement of its children. Importantly, the mapping from label to supernetwork takes account of the labelling - the network with label 2 and label 4 are considered distinct..

Given two community structures, we can now find the largest subtree shared between the two structures, by searching for shared labels and examining how many nodes in the original network this community corresponds to. We can remove leaf nodes as appropriate to reduce the sensitivity to small-scale fluctuations. However, when we generate the labelled trees for the 200 members of the c.1 and d.58 families used previously, even when removing the bottom-level leaf nodes (resulting in a tree of average depth 2.3 for c.1 and 1.6 for d.58), we end up with a label-to-graph mapping with 910 unique elements. 100 of these graphs found are shown in Figure 5.11. This indicates an issue with the method - it is extremely sensitive to small changes in the small-scale structure. Changing the community structure of one node will result in a completely different network. The labelling algorithm exacerbates this problem - the different network will

result in completely different labelling. As this is a bottom-up method, this new labelling will propagate all the way to the top of the tree. However, even when a map is maintained with unlabelled nodes (so that e.g. a triangle of nodes labelled 0,0,0 and 1,2,3 would be considered equivalent), 821 unique graphs are found. The issue is therefore one of combinatorics - once a network has a certain number of nodes, the number of possible arrangements becomes too high.

Protein structure networks tend to have a level of community at roughly the length-scale of a domain, followed by communities at the length-scale of a single helix or beta-sheet strand. This makes them especially difficult to deal with using the iterative labelling method developed here. However, the algorithm may still be a useful tool for networks with larger-scale community structure, such as more traditional social networks.

The method could be refined by removing smaller communities from the tree, or by performing a MCS-based method in which the labelling process is “fuzzy” - networks are given the same label if they have above a certain similarity score.



**Figure 5.11:** 100 of the 910 unique networks found using the tree-labelling algorithm given in this section. Each network represents the arrangement of sub-communities within a single community. We see a vast array of different topologies. We see that most nodes here have the label 0, as they are leaf nodes in the dendrogram.



## Chapter 6

# Application of SOAP to binding sites

In this chapter we move from a network-based description of protein structure to one in which each atom is described in terms of its local environment - the relative position and orientation of its neighbours, as described using the SOAP descriptor. We hope to use the SOAP descriptor introduced in Section 2.2 to identify sets of distinct structural features. We initially prove the validity of the method by successfully distinguishing between well-defined patterns of secondary structure. Following this, we explore the applicability of this descriptor to find and define pharmaceutically relevant ligand binding pockets.

Given two structures, and the SOAP descriptors for their atoms, we can then use the GLOSIM similarity score (also introduced in Section 2.2) to calculate the overall similarity, by effectively matching up atoms pairwise between the two structures as to maximise the overall similarity [17]. Defining the SOAP similarity between atoms  $x_i, x_j$  as  $k(x_i, x_j)$ , the overall similarity score between structures  $A$  and  $B$  is given by:

$$K(A, B) = \sum_{i \in A, j \in B} P_{ij} k(x_i, x_j)$$

where  $P_{ij}$  is the permutation matrix. The GLOSIM score was used in Chapter 4 to calculate the similarity between protein fragments. Here we explore GLOSIM as a metric for classifying distinct protein subsets.

## 6.1 Classifying proteins according to secondary structure composition

Given a training set of protein structures made up of two classes, we can use the GLOSIM score as a to classify unseen test structures. For this test case we chose the two most fundamentally distinct type of protein structures - all alpha-helical (SCOP class **a**) and all beta-sheet (SCOP class **b**) proteins.

We perform classification using a support vector machine (SVM) [144] - specifically a kernel-SVM. Given a set of proteins known to be in class **a**, with class label +1, and a set of proteins in class **b**, with class label -1, we calculate the GLOSIM score for every pair of proteins. The SVM then attempts to draw a hyperplane through our dataset, given the matrix of scores, in order to separate the training data into the two classes. Using this hyperplane, we can then classify a test protein into one of the two classes according to the GLOSIM score  $K(A, B)$  between this new protein and the members of the training set. If the new training protein has a very high score when compared to the proteins in class **a**, it will be assigned to class **a**, and similarly for class **b**. This SVM-based visualisation method has been successfully used in [16] on sets of binding and non-binding ligands - here it is used on the proteins themselves to try and reconstruct SCOP classes.

Once we have trained the SVM, the decision function for deciding the class of a new test structure,  $B$ , is:

$$z_B = \sum_A \alpha_A y_A K(A, B) + \beta$$

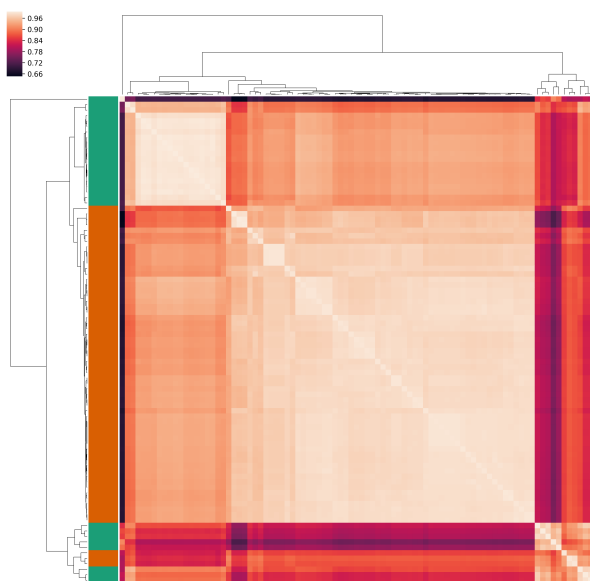
where  $y_A$  is either +1 or -1, the class label of the structure  $A$  in the training set.  $\alpha_A$  and  $\beta$  are the coefficients of the hyperplane, optimised using scikit-learn [137]. The predicted class for  $B$  is the the sign of  $z_B$ , with the magnitude of  $z_B$  roughly corresponding to confidence in the classification.

This method allows us to classify structures according to their structural similarity. Crucially, since the GLOSIM score is simply a linear sum of individual atomic contributions, this decision on class membership can be decomposed into atomic contributions. For atom  $j$  in structure  $B$ , we can obtain the contribution to the SVM classification as:

$$\delta_{Z_J,B} = \sum_A \alpha_A y_A \sum_{i \in A} P_{ij} k_{ij}(A, B) + \frac{\beta}{|B|}$$

where  $|B|$  is the total number of atoms in  $B$ .

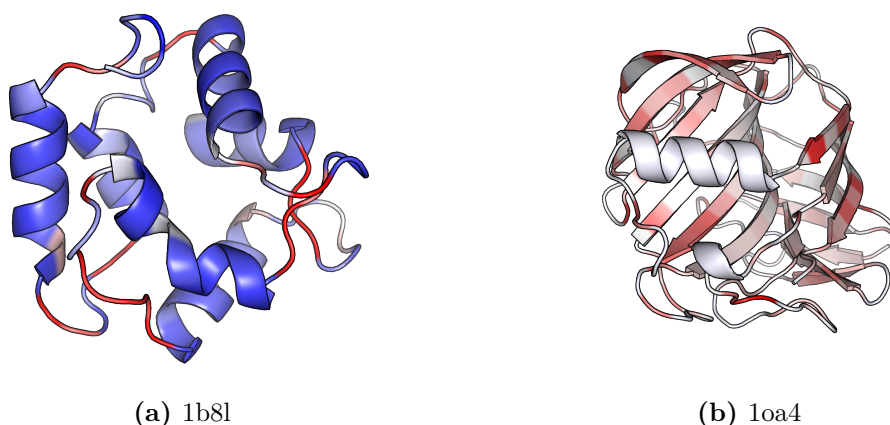
This result allows us to highlight which atoms are key in the classification process. Let us consider our test case of two sets of proteins from SCOP class **a** and **b** - all-alpha helix proteins and all-beta sheet proteins. We perform an all-to-all GLOSIM similarity scoring, using an radial cutoff of  $R = 15$  and smoothing of  $\sigma = 1.5$  as before for the SOAP descriptors, and operating on the alpha-carbons only. We cluster the resulting matrix, as shown in Figure 6.1. We colour the rows of the matrix to indicate whether a protein belongs to class **a** (blue) or **b** (orange), and we see that hierarchical clustering results in good separation between the two classes.



**Figure 6.1:** The clustered matrix of GLOSIM scores for our training data, members of SCOP classes **a** and **b**. A lighter colour in the heatmap indicates a higher similarity score. We colour the rows according to which SCOP class the protein is a member of - blue-green indicates SCOP class **a**, whilst orange indicates SCOP class **b**. We see greater similarity within the classes than between the classes, and hence they are grouped together.

If we then train an SVM on this matrix, we can classify new test proteins, and highlight the contribution of each residue to the classification. We colour a contribution towards scop class **a** (all-alpha) in blue, and a contribution towards scop class **b** (all-beta) in red. Figure 6.2 shows that the classifier behaves as expected - residues corresponding to alpha helices are coloured blue, whilst residues corresponding to beta sheets are coloured red.

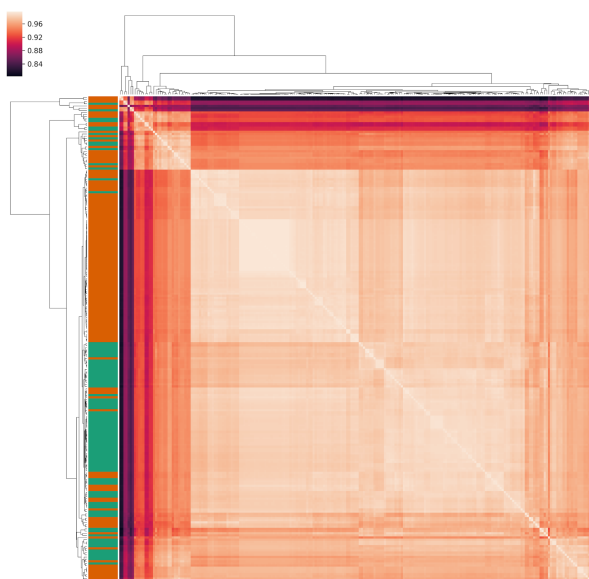
No prior information about secondary structure is provided to the SVM - this information is all learned from the training set. This demonstrates that given a set of proteins known to share structural features, we can use the SVM contributions to highlight the most important atoms or residues.



**Figure 6.2:** Two test proteins from SCOP classes **a** (1b8l) and **b** (1oa4), coloured according to each amino acid's contribution to the SVM classification. Blue indicates that the atom is contributing towards an **a** classification, red a contribution towards class **b**. As expected, the alpha-helices are blue (as class **a** is an all-alpha helix class), and the beta sheets red (as class **b** is an all-beta sheet class).

When we extend this analysis to a test set based on the folds defining **c.1** and **d.58** proteins, as used in Chapter 4, in which the class distinction is more subtle, we still see reasonable separation using hierarchical clustering (Figure 6.3). However, the limitations of this method become apparent - the GLOSIM scores between members of different classes are still high.

This is a result of the score's design - the score is made up of the similarities of individual atoms. Since both classes have both alpha-helices and beta-sheets, similar atoms can be found when comparing between classes. Distinguishing features arise only from atoms at the interfaces between these secondary structure elements, as it is only their relative configuration which distinguishes the folds. The high inter-class GLOSIM score makes classification difficult, as the effective range is reduced.



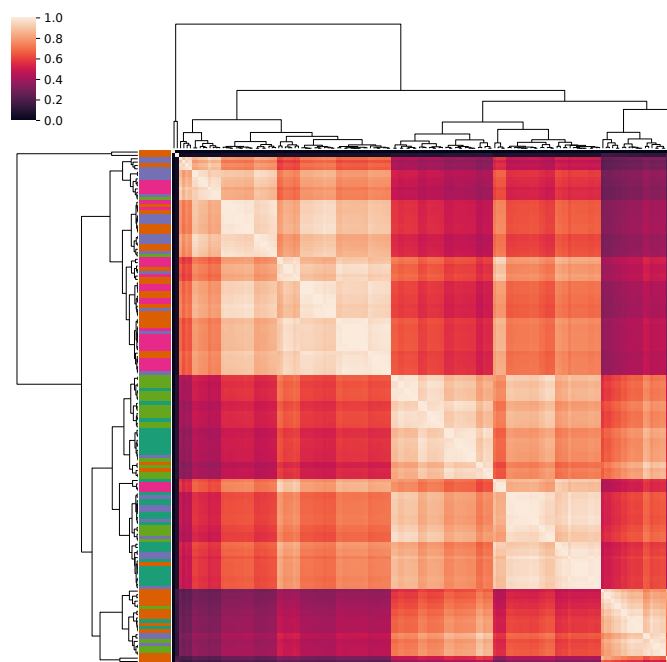
**Figure 6.3:** The clustered matrix of GLOSIM scores for our training data, members of SCOP classes c.1 and d.58. A lighter colour in the heatmap indicates a higher similarity score, with rows coloured according to class membership as before (class c in blue-green and class d in orange). We see that the inter-class similarity score is high, which may be the reason for the failure to fully separate the classes.

## 6.2 Application to Binding Pockets

Given the solid performance on well-defined structural classes, we hoped that the same method could be used to investigate more interesting structural features - for instance, given two sets of proteins which bind different ligands, can we use the contributions to an SVM decision to highlight the binding pockets?

We use the Enzyme Commission (EC) numbering system [145] to select proteins from the high-resolution ASTRAL dataset from Chapter 4, choosing proteins which catalyse five different chemical reactions. We select 30 proteins for each reaction. These proteins should all catalyse the same reaction, and so share common binding sites. We perform the all-to-all GLOSIM scoring on these proteins, and hierarchically cluster and classify as before (Figure 6.4).

We see that while the clustering process yields strong clusters, they fail to map to the EC classes. The related process of SVM classification yields poor results with overall accuracy of 53%. As previously, the clustering and classification process take place using the GLOSIM score between the full proteins - each atomic similarity is weighted equally. As such, the signal due to any similar subregions related to ligand binding is overwhelmed by comparisons between the bulk protein structure, such as defined



**Figure 6.4:** The clustered matrix of GLOSIM scores for a set of 150 protein binders, belonging to 5 EC numbers - 3.5.2.6 (Beta-lactamase, in pink), 3.2.1.8 (xylanase, in purple), 5.2.1.8 (Peptidylprolyl isomerase, in light green), 1.15.1.1 (Superoxide dismutase, in dark green), and 2.7.11.1 (serine/threonine protein kinase, in orange). A lighter colour in the heatmap indicates a higher similarity score, with rows coloured according to class membership as before. We see that the inter-class similarity score is high in some cases, and this may be the reason for the failure to fully separate the classes.

alpha-helix and beta-sheet content. Since the SVM has a poor classification accuracy, the atomic contributions to the SVM are uninformative, and the visualisation process fails.

In the remainder of the chapter, we bypass the SVM contributions as a tool for exploring each atom's importance, and use the GLOSIM data directly to compare a test protein to a previously defined protein set. The GLOSIM comparison process generates two matrices;  $[P_{ij}]$ , the permutation matrix mapping the atom  $i$  in structure  $A$  to atom  $j$  in structure  $B$ , and  $[k(x_i, x_j)]$ , alternatively written as  $[k_{ij}]$ , the similarity between atoms  $i$  and  $j$ . Given this data, we can extract the regions of structural similarity from a test set of proteins using the following algorithm, summarised in Figure 6.5:

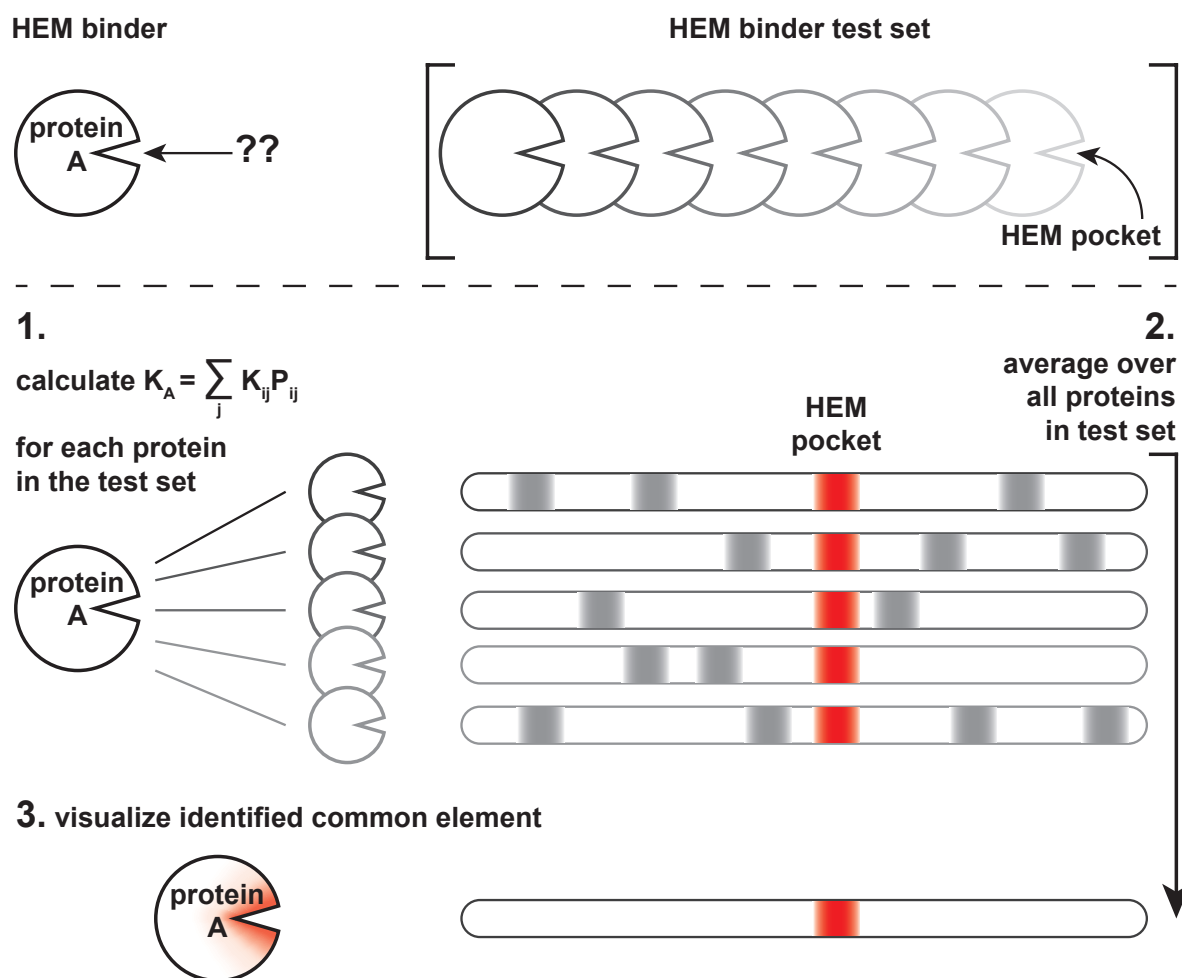
- Take a set of proteins, known to share a structural feature; our test set is comprised of proteins which bind the heme ligand.

- Select a target protein A from that test set, and calculate K and P between that protein and each other protein in the set using REMATCH. (The overall GLOSIM score would then be  $\sum_{i,j} K_{ij}P_{ij}$ ).
- Calculate  $K_A = \sum_j K_{ij}P_{ij}$  for each protein in the set. Each  $K_A$  will be a 1D vector of length N, where N is the number of atoms in protein A, giving the similarity of each atom in A to each protein in the test set.
- Create a matrix in which each row is the  $K_A$  between the target protein and each protein in the test set.
- Average over columns of this matrix, giving the average similarity of each atom in A to the corresponding atoms in the test set,  $\bar{K}_A$ .

This  $\bar{K}_A$  will give the average similarity of each atom in protein A to the corresponding atoms in the test set. The averaging over the atoms in the test set reduces noise due to coincidental atomic similarities, and allows the feature common to the entire test set to be highlighted. This may highlight the binding pocket or important allosteric regions in the case of ligand binders, but could generalise to any set of proteins in which we believe there to be an important subset of the atoms whose relative position is structurally conserved (e.g proteins with a shared GO term, or members of a SCOP superfamily).

To test this hypothesis, we use a curated set of heme binders that have previously been used in protein binding classification [146]. We filter this set to include only those with a ligand with id ‘HEM’ (as alternative confirmations of heme exist), For reasons of computing power, a sample of 100 proteins from the set are chosen. We generate the  $[P_{ij}]$  and  $[k_{ij}]$  matrices using GLOSIM with radial cutoff of  $R = 10$  and  $\sigma = 1.0$ , and for this analysis, we apply GLOSIM to all the atoms in the test proteins. Whilst this increases the computational cost, we require an atomic level of detail in order to investigate the binding pocket.

As shown in Figure 6.6 and Figure 6.7, the results are mixed, and hard to interpret. It is not the case that the binding pocket uniformly receives a high similarity score, and the bulk pocket receives a low similarity score. However, we do see a concentration of high similarity scores around the pocket.

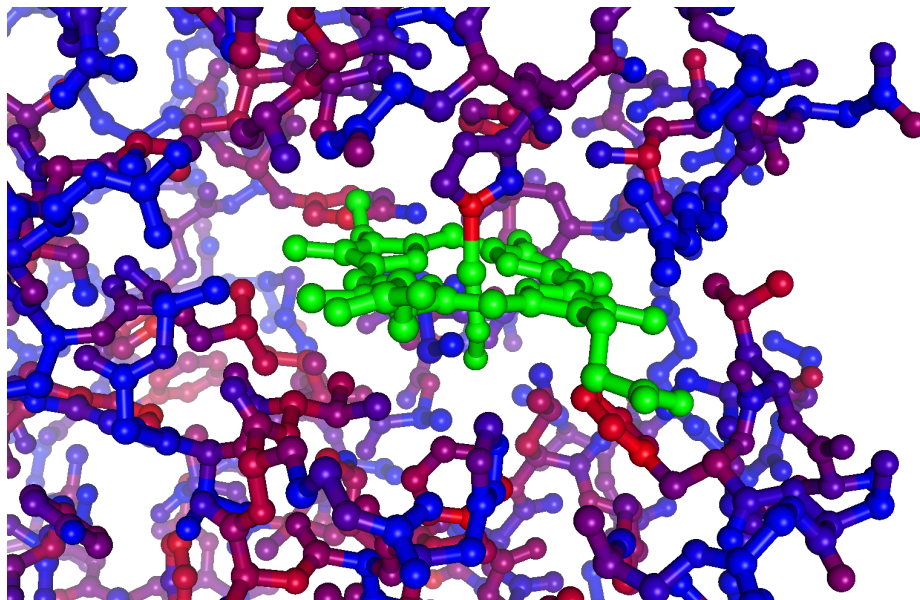


**Figure 6.5:** A schematic of the structure-highlighting process. Given a test set of proteins that we know shares a certain structural motif, we select one protein from the set. We then calculate the GLOSIM matrices giving the atomic similarity for every protein in the set, and sum to give the similarity of each atom in the selected protein to the atoms in the set. By averaging over the set, we hope to eliminate noise, and highlight the common structural feature - in this case, a binding pocket.

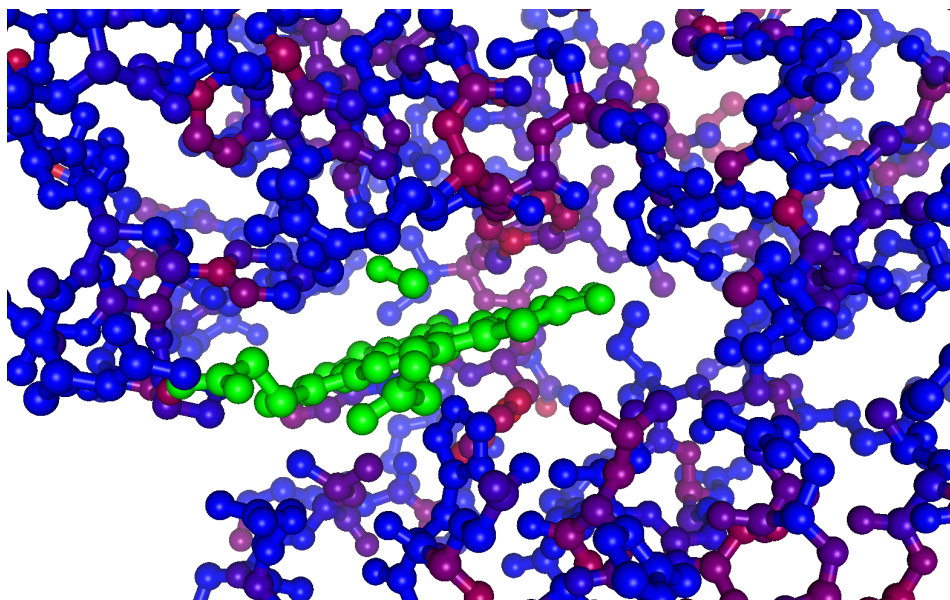
A limitation of this method lies in its tendency to identify isolated sidechain non-carbon atoms as locally similar. Since these side-chain atoms are common on the protein's structure it is highly likely that a match is made between the sidechain on the test protein and every other protein in the set (Figure 6.8). This distorts the result.

A potential way to mitigate this problem would be to only consider the alpha-carbons, rather than each atom individually. However, for binding pocket identification, we found that the resolution of the method was insufficient (data not shown). Instead, we attempt to correct for this problem by performing spatial averaging over the protein, weighted

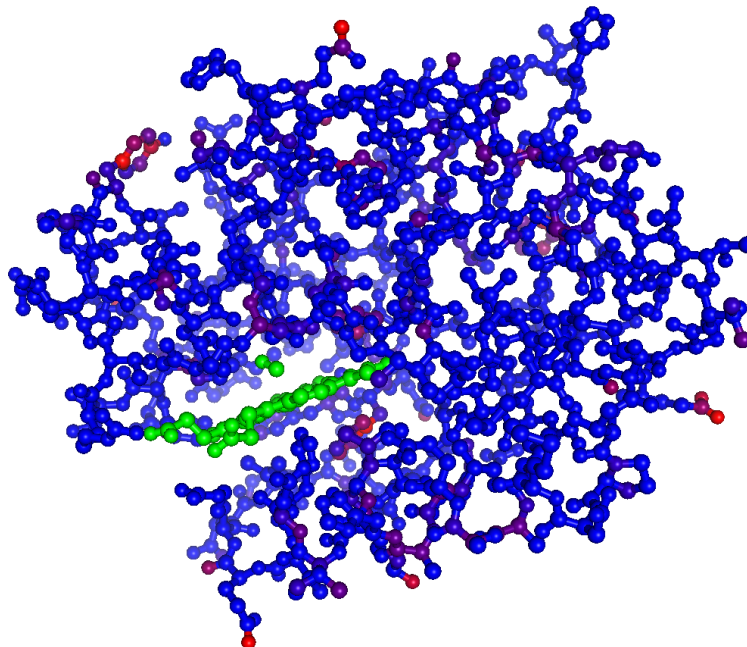




**Figure 6.6:** The results of the similarity-highlighting algorithm on the protein with reference 2D2M, chain A. The protein is coloured by similarity, with blue indicating low similarity, and red high similarity. The HEM ligand is coloured in green. Whilst there is substantial noise in the colouring, we see a concentrated region of similarity near the pocket.



**Figure 6.7:** The results of the similarity-highlighting algorithm on the protein with reference 1X9F, chain D. Blue indicates low similarity, red high similarity, with the HEM ligand coloured green. As in Figure 6.6, the binding pocket is not uniformly coloured, but key atoms are highlighted.



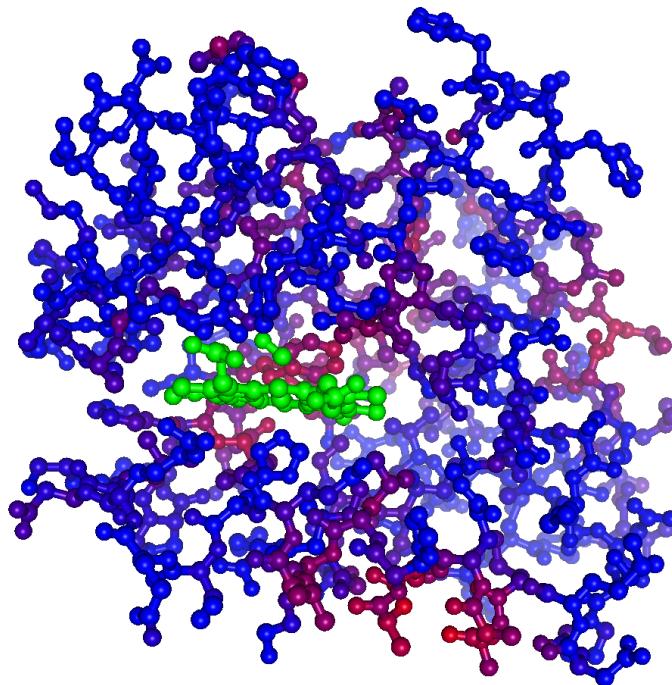
**Figure 6.8:** The similarity-highlighting algorithm acting on 1X9F, chain D, as in Figure 6.7, but showing the full protein. We see that in addition to the atoms close to the HEM ligand, the isolated sidechain non-carbon atoms on the protein’s surface are also given high scores.

by proximity. Given the original scores  $\beta$ , and the  $k$  nearest neighbours of an atom, the new score for atom  $i$ :

$$\hat{\beta}_i = \frac{1}{k} \sum_k \beta_k \exp \left( \frac{-|x_i - x_k|}{\sigma} \right)$$

Here we choose  $k = 20$  and  $\sigma = 10\text{\AA}$ , and  $|x_i - x_k|$  is the distance between atom  $i$  and  $k$ . This gives higher scores to regions of similar atoms, in addition to smoothing the data (Figure 6.9).

We can quantify the extent to which similar atoms are close to the ligand by generating a receiver operating characteristic (ROC) curve [147]. This graph plots the false positive rate (FPR) against the true positive rate (TPR) for a classifier, for a range of model parameters, and is widely used in machine learning to assess model performance. The TPR is given by the number of positive results correctly classified, as a proportion of the total positive results, and the FPR the number of negative results incorrectly classified as positive, as a proportion of the total negative results. A classifier based on random guesses

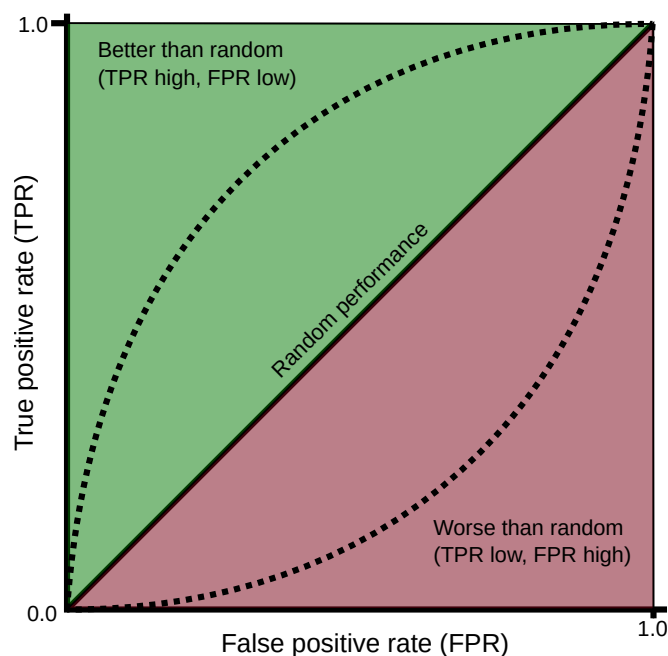


**Figure 6.9:** The similarity-highlighting algorithm acting on 1X9F, chain D, as in Figure 6.7, now with  $k$ -nearest neighbour distance-weighted averaging, with  $k = 20$  and  $\sigma = 10$ . We see that the high-scoring sidechains now have low similarity, and a band of similar structure is revealed around the binding site.

will have the TPR equal to the FPR for the full range, while a near-perfect classifier will have a TRP roughly equal to 1, when the FPR is roughly equal to 0. Figure 6.10 gives an example ROC curve, showing random performance, worse-than-random performance, and better-than-random performance.

We create a classifier from our set of similarity scores as follows:

1. Label each atom as either "High similarity" or "Low similarity" by applying a threshold. Here a similarity threshold of 0.9 is used. These are then our "expected" labels.
2. We create a model in which all the atoms within a certain distance,  $d$ , of the ligand, are assigned the "High similarity" label. In an idealised case, all the atoms in the binding pocket, near the ligand, would be highly similar, with no high-scoring atoms outside the pocket.

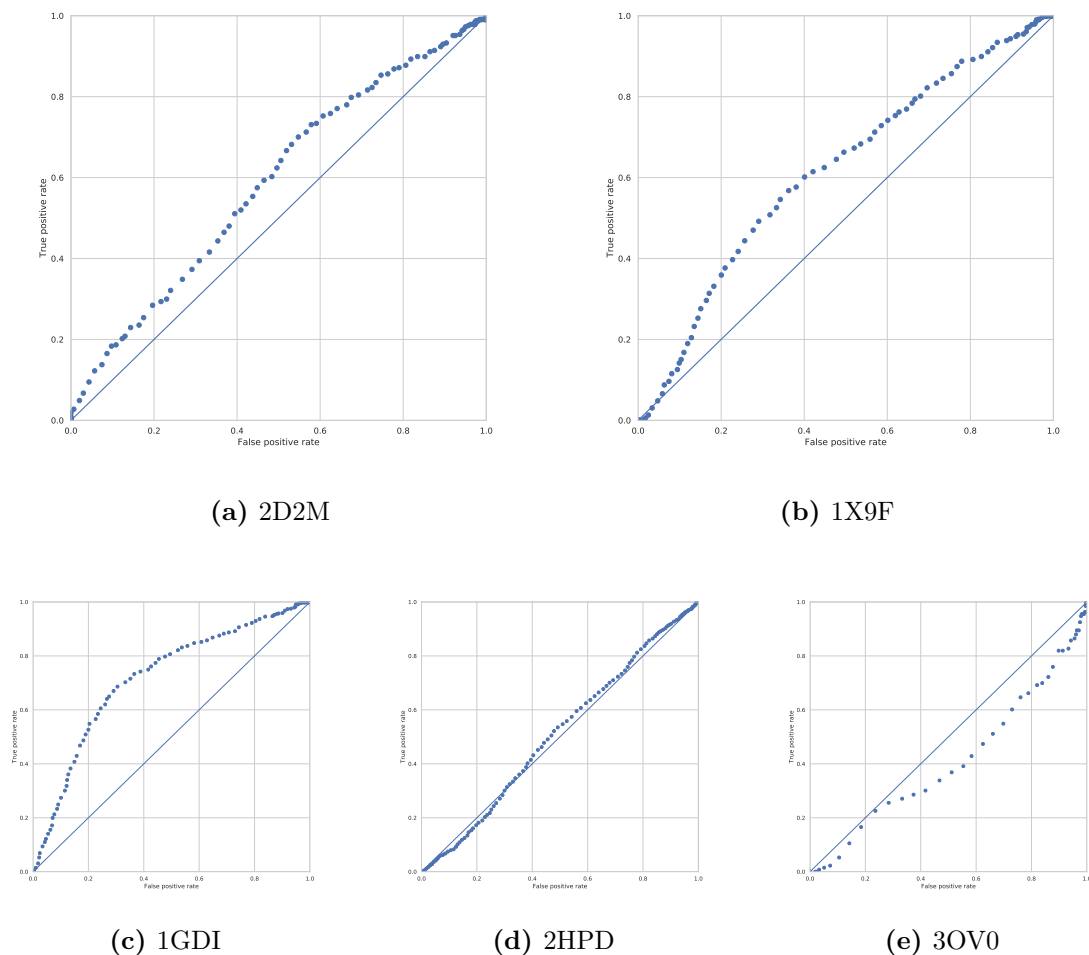


**Figure 6.10:** A demonstration of the ROC curve. The model under test will typically have a parameter that biases the model towards finding positive results. As this parameter is varied, the true positive rate will increase (as more positives are correctly identified), but so will the false positive rate (as more negatives are classified as being positive). In a random classifier, these two rates of increase are the same.

3. The TPR of the model is then the number of atoms that we correctly predict are highly similar using our model, as a fraction of the total number of highly similar atoms. The FPR is defined similarly.
4. As we increase  $d$ , the model predicts that more and more atoms will be highly similar, until all atoms in the protein are labelled as high similarity - at this point the TPR is 1 (as every truly high-similarity atom will be classified as such), as is the FPR (as every low-similarity atom is also classified as highly similar).

The ROC curves for selected proteins in the dataset are shown in Figure 6.11. We see high variability in the quality of the results, with the majority of proteins explored having a better-than-random performance, but with some proteins significantly underperforming, and others doing no better than chance.

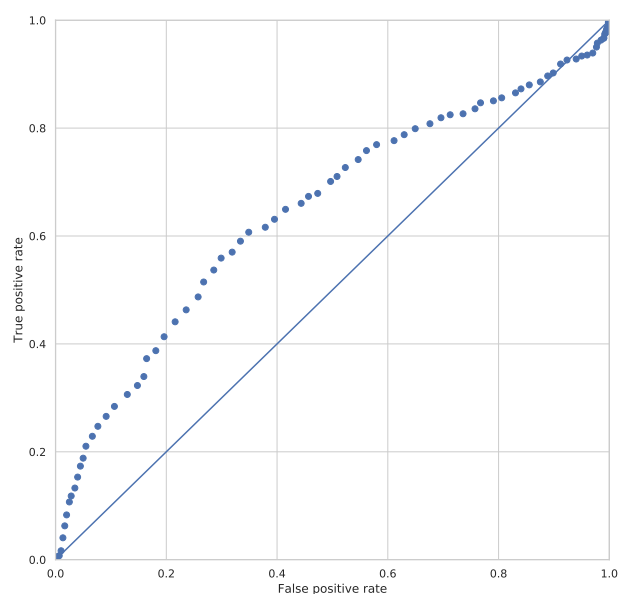
We can also average these curves over the full dataset of 100 heme binders, to get an idea of the extent to which, on average, atoms highlighted using our method are located close to the HEM ligand (Figure 6.12). We see that, on average, the scoring algorithm given here does favour the atoms that are closer to the ligand, as expected.



**Figure 6.11:** The ROC curves for the two proteins explored previously (1X9F and 2D2M), and three other examples (1GDI, 3OV0, and 2HPD). The plotted line indicates the performance expected by chance. We see better than random performance for 1X9F and 2D2M, and good performance for 1GDI. However, the case of 2HPD the method performs no better than chance, and in the case of 3OV0 the method actually underperforms the random classifier.

Overall, this method shows strong promise as an extremely general way to identify key structural features in a dataset. In the case of protein binding, it may highlight the binding site, but it may also reveal key shared regions for allostery.

In fact, the test set of HEM binders (haemoglobins) already used in this work would lend themselves well to the future investigation of binding sites and allosteric elements, as the haemoglobin tetramer is a textbook case of allostery [27].



**Figure 6.12:** The ROC curve, averaged over the full dataset. We see that our scoring algorithm is indeed favouring atoms closer to the ligand, despite the variation in performance shown in Figure 6.11.

# Chapter 7

## Discussion

The aim of this work was to investigate the merit of two new ways of describing a protein structure - one using networks, and one using SOAP descriptors. We found applications for both of these methods, with the features found by both methods comparing well to pre-existing bioinformatic annotation on large protein sets.

The challenge in this work was twofold. On one hand it was necessary to tailor existing network science and condensed-matter approaches, to find de-novo features in large sets of complex protein structures. On the other hand, to enable evaluation of the novel methods presented here, and interface with existing bioinformatics, we also had to find ways to define meaningful datasets, by collating current protein sequence, structure and functional annotation.

We developed a set of necessary tools, in which we integrated a wide variety of protein annotation schemes in one centralised database, provided a uniform interface to those annotation schemes, and included other relevant information such as the AEROSPACI score. These will be of great benefit to any future work developing computational protein structure analysis approaches.

The application of Infomap to protein networks outperformed previous network-based methods, and the hierarchical community structure was applicable at multiple length scales:

- At the domain level, we find good correspondence between Pfam and SCOP domains, extracting highly conserved subregions from a test set of proteins known to have conserved structure.

- At the sub-domain level, we show that communities on the length scale of amino acids are the only ones which result in a significantly compressed description of the protein.
- At the super-domain level, we show that grouping proteins according to their community structure is functionally significant. We also develop a tool for analysing this structure which, though ultimately unapplicable to the types of communities found in protein structure networks, suggests a novel way to compare hierarchical community structure.

The SOAP-based descriptor was successfully extended from its initial use on small molecules to large-scale protein structure analysis, reliably recognising secondary structure elements, and successfully grouping  $\alpha$ -helical and  $\beta$ -sheet containing proteins. For finding binding pockets, the SOAP-based descriptor shows promising initial results, with obvious future avenues for exploration.

While both methods yield statistically significant results, they do not perfectly match manually curated annotation. While the descriptors might serve as a basis of novel protein classification, or drug discovery, we must carefully consider on a case-by-case basis the data sets analysed, and the parameters used, as well as manual curation of the results. Whilst on average the SOAP descriptor will highlight atoms near a HEM ligand, given a dataset of HEM binders, it cannot guarantee a clear result. Similarly, whilst Infomap does produce protein fragments which are highly structurally similar, it may not recapitulate the exact SCOP domain, or structurally conserved region.

Overall, this work demonstrates the merit of applying modern computational methods to the outcomes of systematic biological research. The interface between computing and biology is vast, and in the future the integration of computation methods will become ever more necessary in biological research.



# Appendix

## A.1 Scaling of the number of partitions with network size

Here it is shown that the number of possible partitions grows at least exponentially with the number of nodes, for the simplest possible case of a bisection into two communities. A graph with  $N$  nodes is partitioned into communities with  $N_1$  and  $N_2$  nodes such that  $N_1 + N_2 = N$ .

Then the number of possible partitions is  $\frac{N!}{N_1!N_2!}$ . Using Stirling's approximation and cancelling factors of  $e$  this becomes:

$$\frac{N^{N+1/2}}{N_1^{N_1+1/2} N_2^{N_2+1/2}}$$

For communities of roughly equal size such that  $N_1 \sim N_2 \sim N/2$ , this becomes:

$$\frac{N^{N+1/2}}{\frac{N^{N+1}}{2}} = \frac{2^{N+1}}{\sqrt{N}}$$

For the case of a partition into  $k$  communities, the total number of options generalises to the Stirling number of the second kind [148],  $\left\{ \begin{smallmatrix} N \\ k \end{smallmatrix} \right\}$ . This is defined as:

$$\left\{ \begin{smallmatrix} N \\ k \end{smallmatrix} \right\} := \frac{1}{k!} \sum_{i=0}^k (-1)^i \binom{k}{i} (k-i)^N$$

The number of communities a network can be partitioned into ranges from 1 to  $N$ , so the total number of possible partitions is given by Bell's number:

$$B_N = \sum_{k=1}^N \left\{ \begin{matrix} N \\ k \end{matrix} \right\}$$

## A.2 The AFG algorithm

As discussed in Section 2.4, modularity optimization suffers from a fundamental resolution limit. If the communities are below a certain size, then they will not be detected, even for the limiting case of two cliques separated by a single edge. This limit can be shown to be:

$$\sum_i k_i^{int} < 2 \left( \sqrt{\frac{w}{2}} - 1 \right)$$

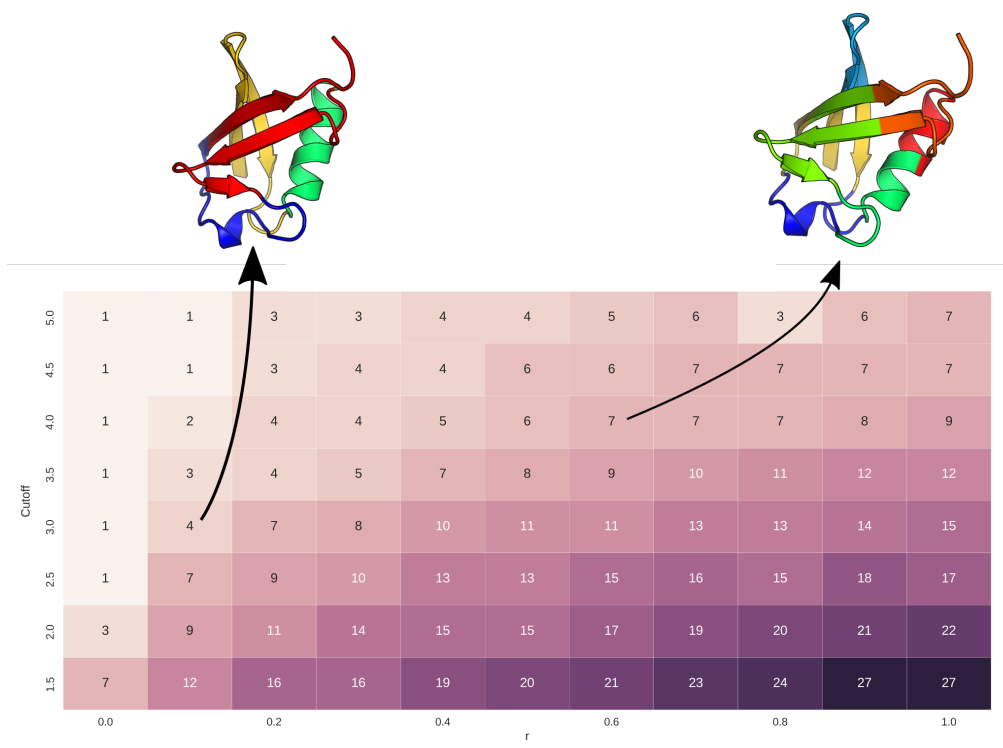
Where  $k_i^{int}$  is the number of edges connecting node  $i$  to other nodes in its community, and  $w$  is the total number of edges in the network, as previously, and the sum is over all nodes within the community. The AFG algorithm uses this resolution limit to investigate the community structure at multiple length scales. By applying self-edges of strength  $r$  to each node, the resolution limit becomes:

$$\sum_i k_i^{int} < \sqrt{2w + Nr} - N_S r - 2$$

As  $\sqrt{r}$  grows more slowly than  $r$ , this allows the resolution limit to be tuned to find smaller or larger modules. Performing a modularity optimisation at each required  $r$  value will give the community structure at multiple scales, with each optimisation an independent process.

To perform each modularity optimisation, a Relax-and-Shake (RASH) algorithm was used. This optimisation method has previously been used to find decompositions of crystal structures [138] and relies on repeated local optimisations, followed by random noise to escape local minima. For details of the algorithm see overleaf or [138].

The AFG method requires a separate modularity optimisation for each community length-scale; as such, it may take weeks or months of CPU time to achieve a good mapping of the community structure. The relevant communities must then be chosen by selecting regions in which the number of communities is stable. This hand-selection makes the AFG method unsuitable for large sets of proteins. However, Figure A.1 shows that the method does achieve sensible partitions of the protein.



**Figure A.1:** A decomposition of 1UBQ into modules using the AFG method, showing the number of modules generated as a function of both the AFG parameter and the cutoff used. Regions in which the number of modules are constant (shaded with the same colour) are believed to correspond to the relevant partitions; the figure demonstrates the issue with the approach, as locating these regions cannot be easily automated. Two example partitions are shown, mapped onto the protein structure, showing that the method correctly separates the secondary structure elements.

## The Relax and Shake (RASH) algorithm

Here the details of the RASH algorithm are given, following the explanation given in [138].

The algorithm is used to optimise the modularity, by changing the community membership of the nodes. This is achieved by a series of local optimisations (relax) followed by shifting a subset of nodes into random communities (shake). This shaking is required due to the high degeneracy of the modularity [93]. The local optimisation follows existing work on modularity optimisation by simulated annealing [149]; for each node, the modularity change resulting from a move into the community of each of its neighbours is calculated. The move resulting in the greatest change is then carried out (if it results in a nett increase). This is repeated until there are no local moves which increase the modularity. A subset of the nodes are then shaken into new communities, and the local optimisation repeated. This continues until 200 consecutive relax-and-shake iterations fail to improve the modularity.

The modularity change as a result of merging any two communities is then calculated; if this results in an increased modularity the merge is performed, and the whole relax-and-shake process is repeated.

The repeated set of relax-and-shake steps, followed by the merge check, can be considered a single step of the optimisation. Following this step, a larger subset of the nodes are shaken into random communities, and the full process repeated until 200 iterations fail to improve the modularity.

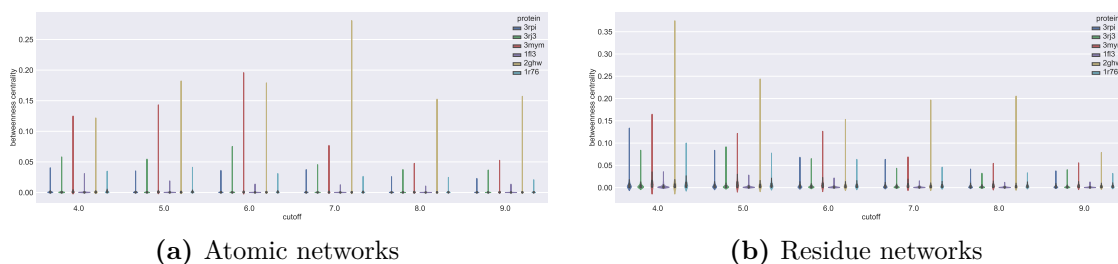
The whole process above is repeated until ten consecutive runs have failed to produce a community structure with a higher modularity.

The degree of repetition at each step is parametrisable, and allows for an accuracy-speed trade-off.

## A.3 Network properties of test proteins

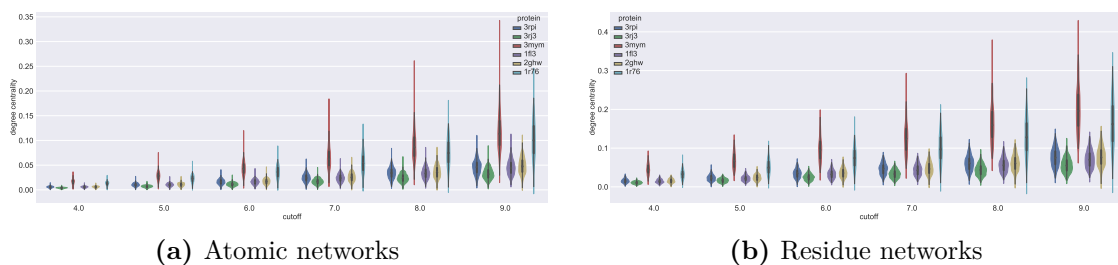
Here we continue the investigation of traditional network properties started in Chapter 3. For an explanation of the network property, please see [53].

**Betweenness centrality:**



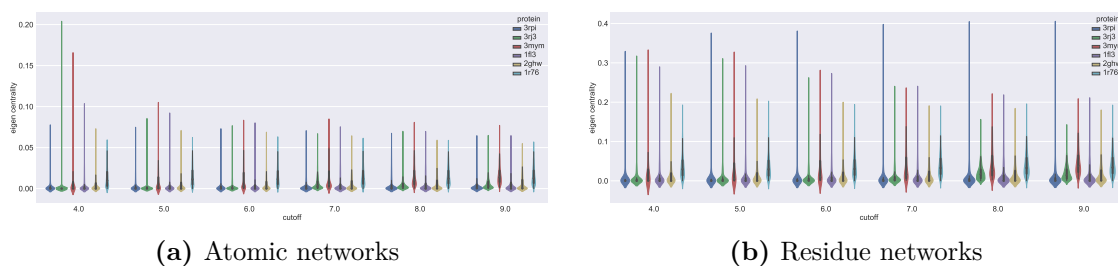
**Figure A.2:** The distributions of betweenness centrality for 6 test proteins.

**Degree centrality:**



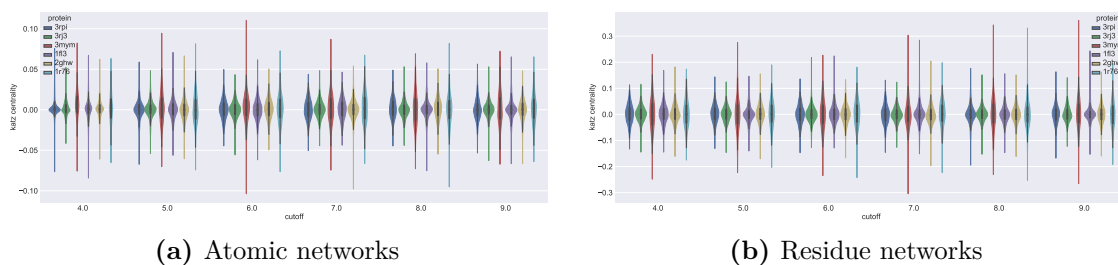
**Figure A.3:** The distributions of degree centrality for 6 test proteins.

**Eigenvector centrality:**



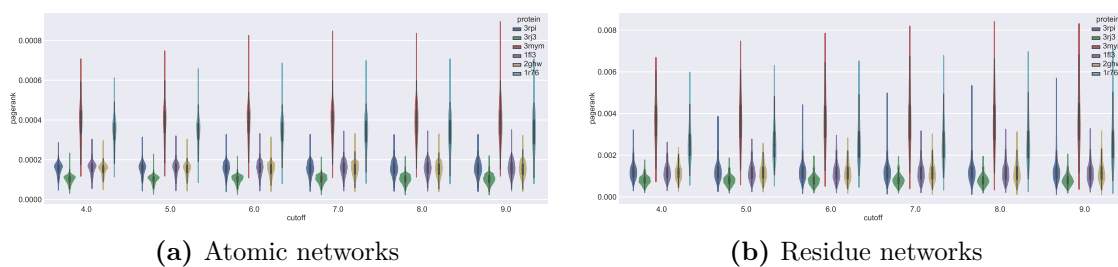
**Figure A.4:** The distributions of eigenvector centrality for 6 test proteins.

### Katz centrality:



**Figure A.5:** The distributions of Katz centrality for 6 test proteins.

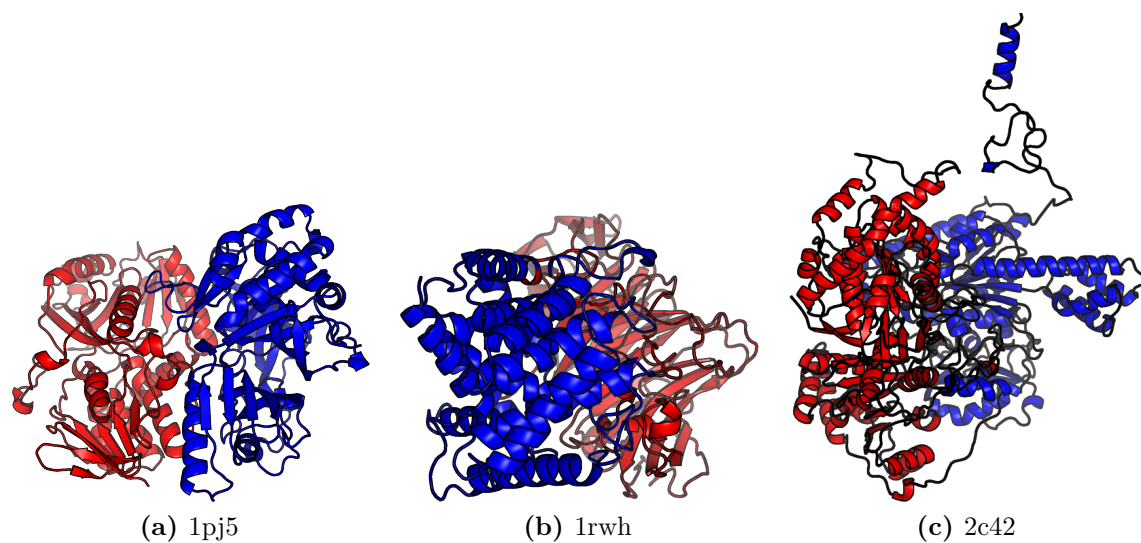
### Pagerank:



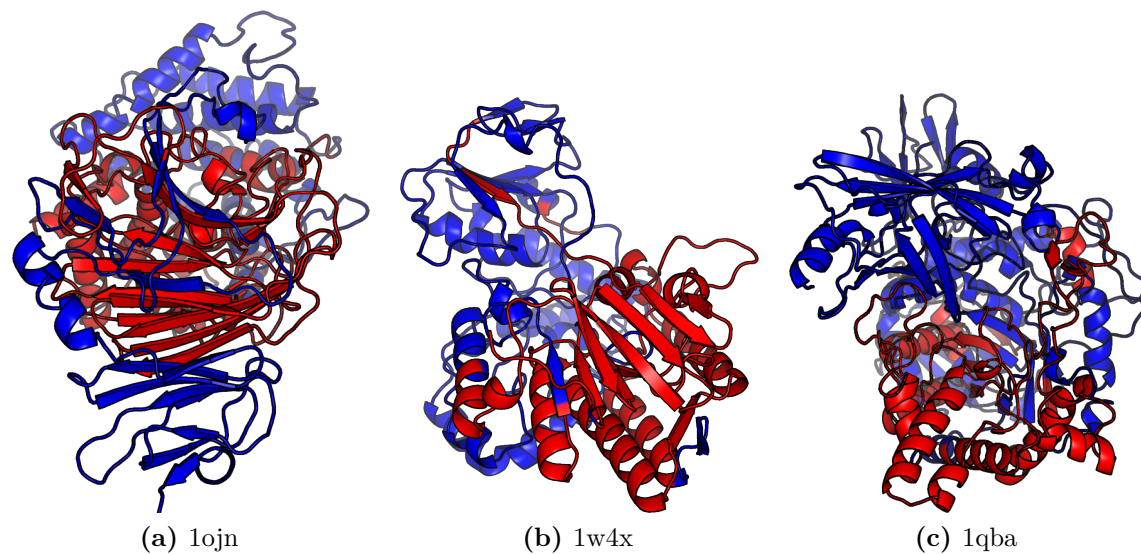
**Figure A.6:** The distributions of Pagerank scores for 6 test proteins.

## A.4 Example communities for the filtered ASTRAL dataset

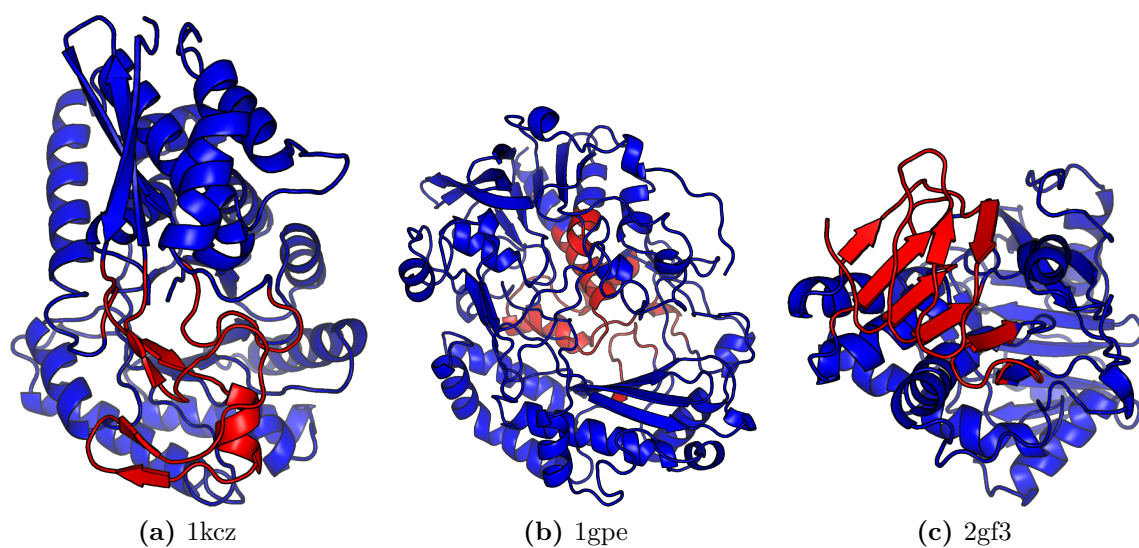
Here we give example fragments for 10 of the 11 clusters found in Section 4.5 - the 11th cluster corresponds to a malformed PDB file and is not shown.



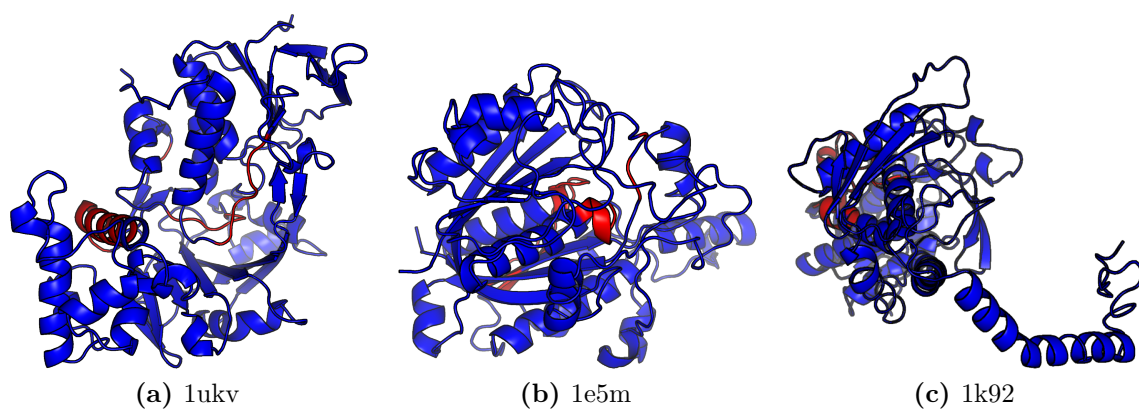
**Figure A.7:** Selected protein fragments from the multi-domain test set, in cluster 0, demonstrating underpartitioning.



**Figure A.8:** Selected protein fragments from the multi-domain test set, in cluster 1. We see correctly sized fragments, but the fragments are not contiguous regions of the protein's structure.

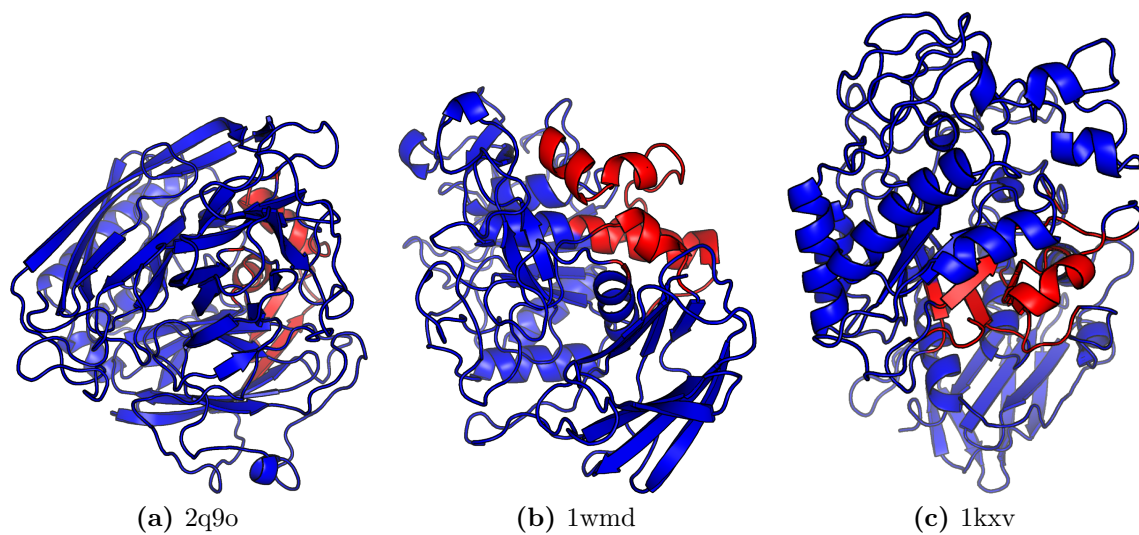


**Figure A.9:** Selected protein fragments from the multi-domain test set, in cluster 2, demonstrating overpartitioning, and showing that dissimilar fragments may still give high GLOSIM scores (notice 2gf3 is an all-beta fragment whilst 1gpe is all-alpha).

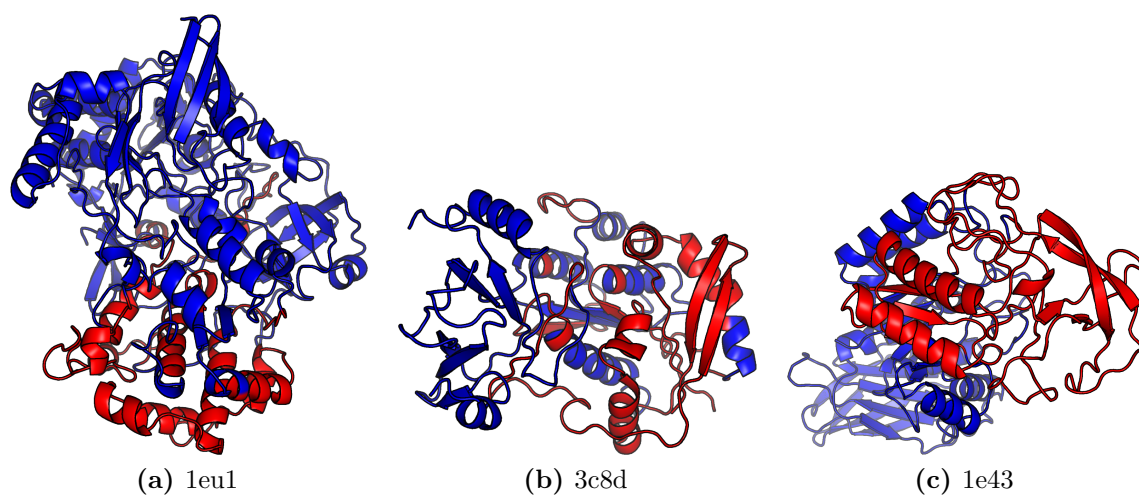


**Figure A.10:** Selected protein fragments from the multi-domain test set, in cluster 3, again showing overpartitioning.

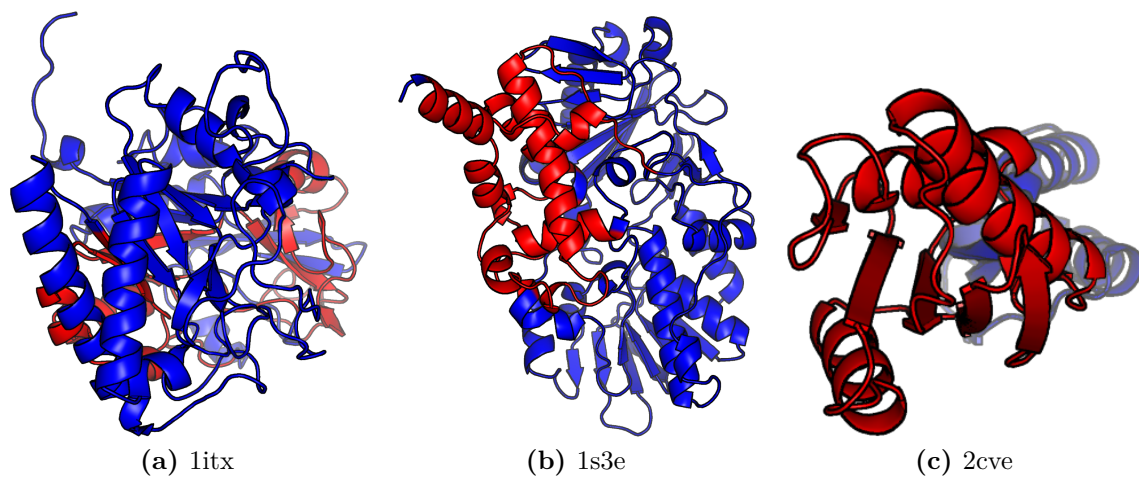




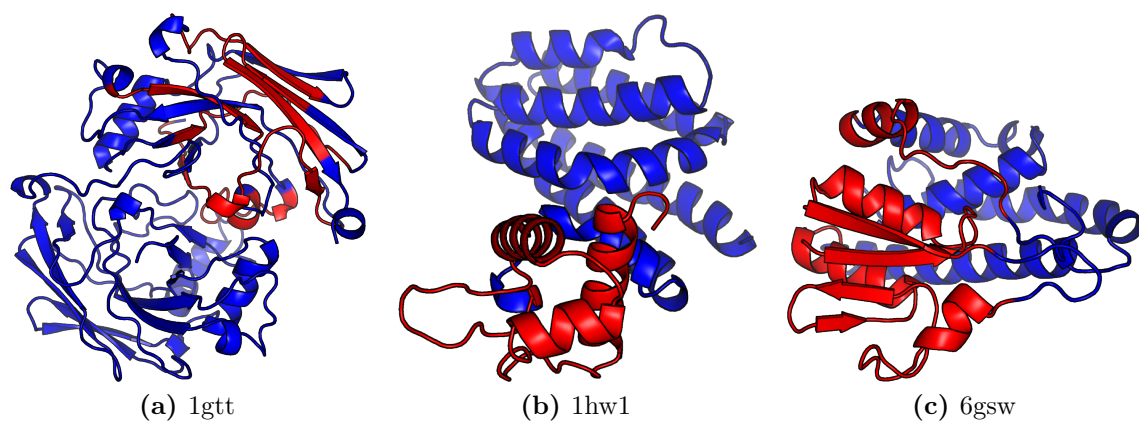
**Figure A.11:** Selected protein fragments from the multi-domain test set, in cluster 4



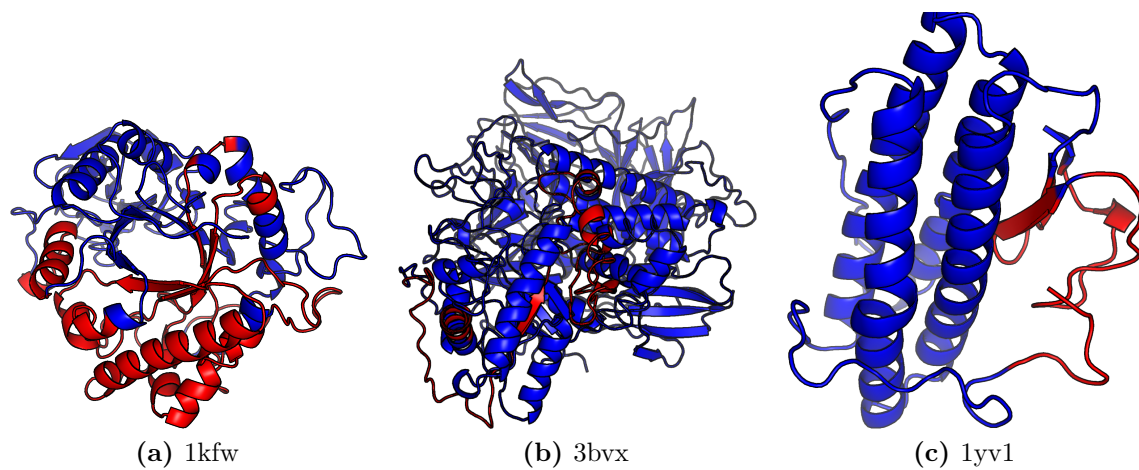
**Figure A.12:** Selected protein fragments from the multi-domain test set, in cluster 5



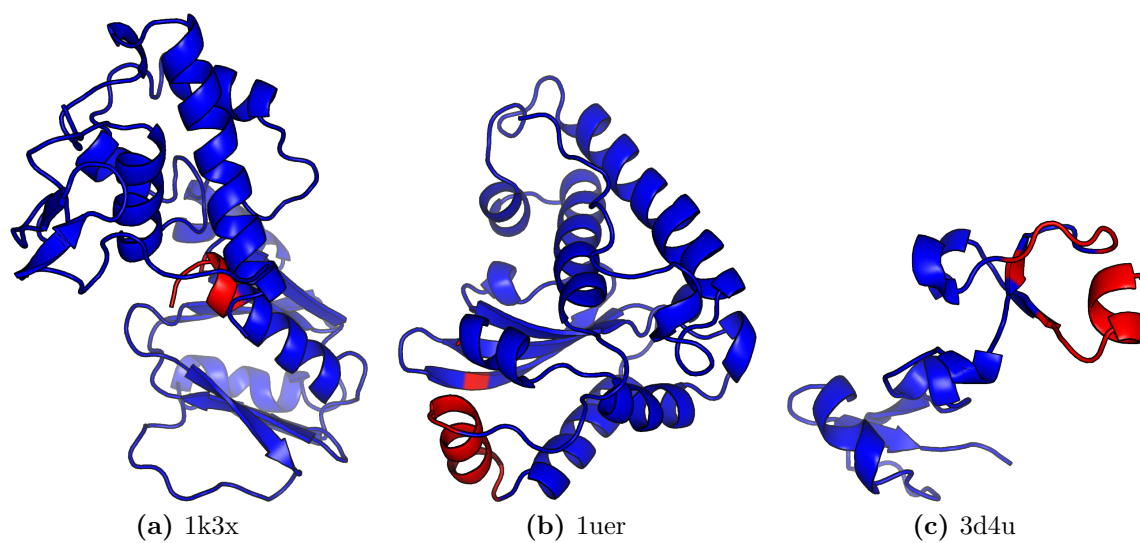
**Figure A.13:** Selected protein fragments from the multi-domain test set, in cluster 6



**Figure A.14:** Selected protein fragments from the multi-domain test set, in cluster 7



**Figure A.15:** Selected protein fragments from the multi-domain test set, in cluster 8



**Figure A.16:** Selected protein fragments from the multi-domain test set, in cluster 9

# Bibliography

- [1] H. M. Berman, B. Coimbatore Narayanan, L. D. Costanzo, S. Dutta, S. Ghosh, B. P. Hudson, C. L. Lawson, E. Peisach, A. Prlić, P. W. Rose, et al., Trendspotting in the protein data bank, *FEBS letters* 587 (8) (2013) 1036–1045 (2013).
- [2] N. K. Fox, S. E. Brenner, J.-M. Chandonia, SCOPe: Structural classification of proteins-extended, integrating scop and astral data and classification of new structures, *Nucleic acids research* 42 (D1) (2013) D304–D309 (2013).
- [3] H. Cheng, R. D. Schaeffer, Y. Liao, L. N. Kinch, J. Pei, S. Shi, B.-H. Kim, N. V. Grishin, ECOD: an evolutionary classification of protein domains, *PLoS computational biology* 10 (12) (2014) e1003926 (2014).
- [4] M. Girvan, M. Newman, Community structure in social and biological networks, *P. Natl. Acad. Sci. Usa* 99 (12) (2002) 7821–7826 (Jun 11 2002). doi:10.1073/Pnas.122653799.
- [5] S. Fortunato, Community detection in graphs, *Phys. Rep.* 486 (3-5) (2010) 75–174 (Feb 2010). doi:10.1016/J.Physrep.2009.11.002.
- [6] S. Fortunato, D. Hric, Community detection in networks: A user guide, *Physics Reports* 659 (2016) 1–44 (2016).
- [7] M. T. Schaub, J.-C. Delvenne, M. Rosvall, R. Lambiotte, The many facets of community detection in complex networks, *Applied Network Science* 2 (1) (2017) 4 (2017).
- [8] L. A. Adamic, N. Glance, The political blogosphere and the 2004 us election: divided they blog, in: *Proceedings of the 3rd international workshop on Link discovery*, ACM, 2005, pp. 36–43 (2005).
- [9] R. Guimera, S. Mossa, A. Turtshi, L. Amaral, The worldwide air transportation network: Anomalous centrality, community structure, and cities’ global roles, *P. Natl. Acad. Sci. Usa* 102 (22) (2005) 7794–7799 (May 31 2005). doi:10.1073/Pnas.0407994102.
- [10] G. Flake, S. Lawrence, C. Giles, F. Coetzee, Self-organization and identification of web communities, *Computer* 35 (3) (2002) 66+ (Mar 2002). doi:10.1109/2.989932.
- [11] P. Gleiser, L. Danon, Community structure in jazz, *Adv. Comp. Sys.* 6 (4) (2003) 565–573 (Dec 2003). doi:10.1142/S0219525903001067.
- [12] A. Del Sol, M. J. Araúzo-Bravo, D. Amoros, R. Nussinov, Modular architecture of protein structures and allosteric communications: potential implications for signaling proteins and regulatory linkages, *Genome biology* 8 (5) (2007) R92 (2007).
- [13] L. Di Paola, A. Giuliani, Protein contact network topology: a natural language for allostery, *Current opinion in structural biology* 31 (2015) 43–48 (2015).

- [14] B. R. Amor, M. T. Schaub, S. N. Yaliraki, M. Barahona, Prediction of allosteric sites and mediating interactions through bond-to-bond propensities, *Nature communications* 7 (2016).
- [15] A. P. Bartók, R. Kondor, G. Csányi, On representing chemical environments, *Phys. Rev. B* 87 (2013) 184115 (May 2013). doi:10.1103/PhysRevB.87.184115.
- [16] A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, M. Ceriotti, Machine learning unifies the modeling of materials and molecules, *Science advances* 3 (12) (2017) e1701816 (2017).
- [17] S. De, A. P. Bartók, G. Csányi, M. Ceriotti, Comparing molecules and solids across structural and alchemical space, *Physical Chemistry Chemical Physics* 18 (20) (2016) 13754–13769 (2016).
- [18] R. Milo, What is the total number of protein molecules per cell volume? a call to rethink some published values, *Bioessays* 35 (12) (2013) 1050–1055 (2013).
- [19] C. B. Anfinsen, Principles that govern the folding of protein chains, *Science* 181 (4096) (1973) 223–230 (1973). doi:10.1126/Science.181.4096.223.
- [20] J. Frydman, Folding of newly translated proteins in vivo: The role of molecular chaperones, *Annual Review of Biochemistry* 70 (1) (2001) 603–647 (2001). doi:10.1146/annurev.biochem.70.1.603.
- [21] F. Crick, Central dogma of molecular biology, *Nature* 227 (5258) (1970) 561–& (1970). doi:10.1038/227561A0.
- [22] M.-S. Kim, S. M. Pinto, D. Getnet, R. S. Nirujogi, S. S. Manda, R. Chaerkady, A. K. Madugundu, D. S. Kelkar, R. Isserlin, S. Jain, et al., A draft map of the human proteome, *Nature* 509 (7502) (2014) 575–581 (2014).
- [23] W. P. Grant, Structural analysis of proteins using community detection, Master’s thesis, University of Cambridge (2016).
- [24] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology Of The Cell*, 4th Edition, Garland Science, 2002 (2002).
- [25] L. Pauling, R. B. Corey, H. R. Branson, The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain, *Proceedings of the National Academy of Sciences* 37 (4) (1951) 205–211 (1951).
- [26] L. Pauling, R. B. Corey, The pleated sheet, a new layer configuration of polypeptide chains, *Proceedings of the National Academy of Sciences of the United States of America* 37 (5) (1951) 251 (1951).
- [27] M. Perutz, Stereochemistry of cooperative effects in haemoglobin: haem–haem interaction and the problem of allostery, *Nature* 228 (5273) (1970) 726 (1970).
- [28] A. Bateman, et al., Uniprot: A hub for protein information, *Nucleic Acids Res.* 43 (D1) (2015) D204–D212 (Jan 28 2015). doi:10.1093/Nar/Gku989.
- [29] P. W. Rose, A. Prlić, A. Altunkaya, C. Bi, A. R. Bradley, C. H. Christie, L. D. Costanzo, J. M. Duarte, S. Dutta, Z. Feng, et al., The RCSB protein data bank: integrative view of protein, gene and 3d structural information, *Nucleic acids research* (2016) gkw1000 (2016).

- [30] J. Shendure, H. Ji, Next-generation dna sequencing, *Nat. Biotechnol.* 26 (10) (2008) 1135–1145 (Oct 2008). doi:10.1038/Nbt1486.
- [31] Nhgri, The cost of sequencing a human genome (2016).  
URL [Http://Bit.Ly/1U15Ydw](http://Bit.Ly/1U15Ydw)
- [32] S. R. Eddy, Profile hidden markov models., *Bioinformatics* 14 (9) (1998) 755–763 (1998).
- [33] R. D. Finn, P. Coghill, R. Y. Eberhardt, S. R. Eddy, J. Mistry, A. L. Mitchell, S. C. Potter, M. Punta, M. Qureshi, A. Sangrador-Vegas, G. A. Salazar, J. Tate, A. Bateman, The Pfam protein families database: Towards a more sustainable future, *Nucleic Acids Res.* 44 (D1) (2016) D279–D285 (Jan 4 2016). doi:10.1093/Nar/Gkv1344.
- [34] A. G. Murzin, S. E. Brenner, T. Hubbard, C. Chothia, SCOP: a structural classification of proteins database for the investigation of sequences and structures, *Journal of molecular biology* 247 (4) (1995) 536–540 (1995).
- [35] I. Sillitoe, T. E. Lewis, A. Cuff, S. Das, P. Ashford, N. L. Dawson, N. Furnham, R. A. Laskowski, D. Lee, J. G. Lees, S. Lehtinen, R. A. Studer, J. Thornton, C. A. Orengo, Cath: Comprehensive structural and functional annotations for genome sequences, *Nucleic Acids Res.* 43 (D1) (2015) D376–D381 (Jan 28 2015). doi:10.1093/Nar/Gku947.
- [36] L. Holm, C. Sander, Parser for protein folding units, *Proteins* 19 (3) (1994) 256–268 (1994). doi:10.1002/Prot.340190309.
- [37] M. B. Swindells, A procedure for detecting structural domains in proteins, *Protein Sci.* 4 (1) (1995) 103–112 (1995). doi:10.1002/Pro.5560040113.
- [38] A. S. Siddiqui, G. J. Barton, Continuous and discontinuous domains: An algorithm for the automatic generation of reliable protein domain definitions, *Protein Sci.* 4 (5) (1995) 872–884 (1995). doi:10.1002/Pro.5560040507.
- [39] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al., Gene ontology: tool for the unification of biology, *Nature genetics* 25 (1) (2000) 25–29 (2000).
- [40] D. W. Huang, B. T. Sherman, R. A. Lempicki, Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists, *Nucleic acids research* 37 (1) (2008) 1–13 (2008).
- [41] S. Y. Rhee, V. Wood, K. Dolinski, S. Draghici, Use and misuse of the gene ontology annotations, *Nature Reviews Genetics* 9 (7) (2008) 509–515 (2008).
- [42] S. Velankar, J. M. Dana, J. Jacobsen, G. van Ginkel, P. J. Gane, J. Luo, T. J. Oldfield, C. O’Á-Donovan, M.-J. Martin, G. J. Kleywegt, Sifts: structure integration with function, taxonomy and sequences resource, *Nucleic acids research* 41 (D1) (2013) D483–D489 (2013).
- [43] I. N. Shindyalov, P. E. Bourne, Protein structure alignment by incremental combinatorial extension (CE) of the optimal path., *Protein Engineering, Design and Selection* 11 (9) (1998) 739–747 (09 1998). doi:10.1093/protein/11.9.739.
- [44] A. Poleksic, Algorithms for optimal protein structure alignment, *Bioinformatics* 25 (21) (2009)

- 2751–2756 (09 2009). doi:10.1093/bioinformatics/btp530.
- [45] Y. Zhang, J. Skolnick, Tm-align: A protein structure alignment algorithm based on the tm-score, *Nucleic Acids Res.* 33 (7) (2005) 2302–2309 (2005). doi:10.1093/Nar/Gki524.
- [46] N. Malod-Dognin, N. Pržulj, Gr-align: fast and flexible alignment of protein 3d structures using graphlet degree similarity, *Bioinformatics* 30 (9) (2014) 1259–1265 (2014).
- [47] L. Holm, C. Sander, Protein structure comparison by alignment of distance matrices, *Journal of molecular biology* 233 (1) (1993) 123–138 (1993).
- [48] I. Eidhammer, I. Jonassen, W. R. Taylor, Structure comparison and structure patterns, *Journal of Computational Biology* 7 (5) (2000) 685–716 (2000).
- [49] H. W. Kuhn, The hungarian method for the assignment problem, *Naval research logistics quarterly* 2 (1-2) (1955) 83–97 (1955).
- [50] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys.* 74 (2002) 47–97 (Jan 2002). doi:10.1103/Revmodphys.74.47.
- [51] M. Newman, The structure and function of complex networks, *Siam Review* 45 (2) (2003) 167–256 (Jun 2003). doi:10.1137/S003614450342480.
- [52] S. N. Dorogovtsev, J. F. Mendes, Evolution of networks, *Advances in physics* 51 (4) (2002) 1079–1187 (2002).
- [53] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, Inc., New York, Ny, USA, 2010 (2010).
- [54] D. J. de Solla Price, Networks of scientific papers, *Science* 149 (3683) (1965) 510–515 (1965).
- [55] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener, Graph structure in the web, *Computer networks* 33 (1) (2000) 309–320 (2000).
- [56] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, *science* 286 (5439) (1999) 509–512 (1999).
- [57] P. Erdős, A. Rényi, On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci* 5 (1) (1960) 17–60 (1960).
- [58] V. Chvátal, The smallest triangle-free 4-chromatic 4-regular graph, *Journal of Combinatorial Theory* 9 (1) (1970) 93–94 (1970).
- [59] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D. U. Hwang, Complex networks: Structure and dynamics, *Phys. Rep.* 424 (4-5) (2006) 175–308 (Feb 2006). doi:10.1016/J.Physrep.2005.10.009.
- [60] D. J. Watts, S. H. Strogatz, Collective dynamics of ‘small-world’ networks, *Nature* 393 (6684) (1998) 440–442 (Jun 1998). doi:10.1038/30918.
- [61] L. Babai, Graph isomorphism in quasipolynomial time, *CoRR* abs/1512.03547 (2015). URL <http://arxiv.org/abs/1512.03547>
- [62] S. Virtanen, Properties of nonuniform random graph models (A77) (2003) 109 (May 2003).

- [63] A. Krause, K. Frank, D. Mason, R. Ulanowicz, W. Taylor, Compartments revealed in food-web structure, *Nature* 426 (6964) (2003) 282–285 (Nov 20 2003). doi:10.1038/Nature02115.
- [64] V. Spirin, L. A. Mirny, Protein complexes and functional modules in molecular networks, *Proceedings of the National Academy of Sciences* 100 (21) (2003) 12123–12128 (2003).
- [65] E. Ravasz, A. Somera, D. Mongru, Z. Oltvai, A. Barabasi, Hierarchical organization of modularity in metabolic networks, *Science* 297 (5586) (2002) 1551–1555 (Aug 30 2002). doi:10.1126/Science.1073374.
- [66] E. Ravasz, A. Barabasi, Hierarchical organization in complex networks, *Phys. Rev. E* 67 (2, 2) (Feb 2003). doi:10.1103/Physreve.67.026112.
- [67] P. Holme, M. Huss, H. Jeong, Subnetwork hierarchies of biochemical pathways, *Bioinformatics* 19 (4) (2003) 532–538 (Mar 1 2003). doi:10.1093/Bioinformatics/Btg033.
- [68] R. Mantegna, Hierarchical structure in financial markets, *Eur. Phys. J. B.* 11 (1) (1999) 193–197 (Sep 1999). doi:10.1007/S100510050929.
- [69] W. W. Zachary, An information flow model for conflict and fission in small groups, *Journal of anthropological research* 33 (4) (1977) 452–473 (1977).
- [70] H. C. White, S. A. Boorman, R. L. Breiger, Social structure from multiple networks. i. blockmodels of roles and positions, *American journal of sociology* 81 (4) (1976) 730–780 (1976).
- [71] P. W. Holland, K. B. Laskey, S. Leinhardt, Stochastic blockmodels: First steps, *Social networks* 5 (2) (1983) 109–137 (1983).
- [72] S. E. Schaeffer, Graph clustering, *Comp. Sci. Rev.* 1 (1) (2007) 27–64 (2007). doi:10.1016/J.Cosrev.2007.05.001.
- [73] M. A. Porter, J.-P. Onnela, P. J. Mucha, Communities in networks, *Notices Amer. Soc. Math.* 56 (9) (2009) 1082–1097 (2009).
- [74] M. Coscia, F. Giannotti, D. Pedreschi, A classification for community discovery methods in complex networks, *Statistical Analysis and Data Mining* 4 (5) (2011) 512–546 (2011).
- [75] S. Parthasarathy, Y. Ruan, V. Satuluri, Community discovery in social networks: Applications, methods and emerging trends, in: *Social network data analytics*, Springer, 2011, pp. 79–113 (2011).
- [76] M. E. J. Newman, Communities, modules and large-scale structure in networks, *Nat. Phys.* 8 (1) (2012) 25–31 (Jan 2012). doi:10.1038/Nphys2162.
- [77] T. Chakraborty, A. Dalmia, A. Mukherjee, N. Ganguly, Metrics for community analysis: A survey, preprint arXiv:1604.03512 (2016).
- [78] A. Pothen, Graph partitioning algorithms with applications to scientific computing, *ICASE LaRC Interdisciplinary Series in Science and Engineering* 4 (1997) 323–368 (1997).
- [79] B. W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, *The Bell system technical journal* 49 (2) (1970) 291–307 (1970).
- [80] L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, *J. Stat. Mech.-Theory E* (Sep 2005). doi:10.1088/1742-5468/2005/09/P09008.



- [81] A. Lancichinetti, S. Fortunato, Community detection algorithms: A comparative analysis, *Phys. Rev. E* 80 (2009) 056117 (Nov 2009). doi:10.1103/Physreve.80.056117.
- [82] S. Harenberg, G. Bello, L. Gjeltrema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, N. Samatova, Community detection in large-scale networks: A survey and empirical evaluation, *Wiley Interdisciplinary Reviews: Computational Statistics* 6 (6) (2014) 426–439 (2014). doi:10.1002/Wics.1319.
- [83] G. K. Orman, V. Labatut, H. Cherifi, Comparative evaluation of community detection algorithms: A topological approach, *J. Stat. Mech.* (Aug 2012). doi:10.1088/1742-5468/2012/08/P08001.
- [84] J. Leskovec, K. J. Lang, M. Mahoney, Empirical comparison of algorithms for network community detection, in: *Proceedings Of The 19th International Conference On World Wide Web*, Acm, 2010, pp. 631–640 (2010).
- [85] F. Moradi, T. Olovsson, P. Tsigas, An evaluation of community detection algorithms on large-scale email traffic, in: *Experimental Algorithms*, Springer, 2012, pp. 283–294 (2012).
- [86] L. Peel, D. B. Larremore, A. Clauset, The ground truth about metadata and community detection in networks, *Science Advances* 3 (5) (2017) e1602548 (2017).
- [87] U. Von Luxburg, A tutorial on spectral clustering, *Statistics and computing* 17 (4) (2007) 395–416 (2007).
- [88] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks, *Phys. Rev. E* 69 (2, 2) (Feb 2004). doi:10.1103/Physreve.69.026113.
- [89] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, *Ieee T. Knowl. Data. En.* 20 (2) (2008) 172–188 (Feb 2008). doi:10.1109/Tkde.2007.190689.
- [90] S. Fortunato, M. Barthelemy, Resolution limit in community detection, *P. Natl. Acad. Sci. Usa* 104 (1) (2007) 36–41 (Jan 2 2007). doi:10.1073/Pnas.0605965104.
- [91] J. M. Kumpula, J. Saramaki, K. Kaski, J. Kertesz, Limited resolution in complex network community detection with potts model approach, *Eur. Phys. J. B* 56 (1) (2007) 41–45 (Mar 2007). doi:10.1140/Epjb/E2007-00088-4.
- [92] L. K. Branting, Information theoretic criteria for community detection, in: L. Giles, M. Smith, J. Yen, H. Zhang (Eds.), *Asonam*, Vol. 5498 of *Lecture Notes In Computer Science*, 2010, pp. 114–130 (2010).
- [93] B. H. Good, Y.-A. De Montjoye, A. Clauset, Performance of modularity maximization in practical contexts, *Phys. Rev. E* 81 (4, 2) (Apr 2010). doi:10.1103/Physreve.81.046106.
- [94] M. E. J. Newman, Equivalence between modularity optimization and maximum likelihood methods for community detection, *Phys. Rev. E* 94 (Nov 2016). doi:10.1103/PhysRevE.94.052315.
- [95] J.-C. Delvenne, M. T. Schaub, S. N. Yaliraki, M. Barahona, The stability of a graph partition: A dynamics-based framework for community detection, in: *Dynamics On and Of Complex Networks*, Volume 2, Springer, 2013, pp. 221–242 (2013).

- [96] B. Karrer, M. E. Newman, Stochastic blockmodels and community structure in networks, *Physical Review E* 83 (1) (2011) 016107 (2011).
- [97] T. P. Peixoto, Nonparametric bayesian inference of the microcanonical stochastic block model, *Phys. Rev. E* 95 (2017) 012317 (Jan 2017). doi:10.1103/PhysRevE.95.012317.
- [98] T. P. Peixoto, Hierarchical block structures and high-resolution model selection in large networks, *Phys. Rev. X* 4 (2014) 011047 (Mar 2014). doi:10.1103/PhysRevX.4.011047.
- [99] P. D. Grünwald, I. J. Myung, M. A. Pitt, *Advances In Minimum Description Length: Theory And Applications*, Mit Press, 2005 (2005).
- [100] C. E. Shannon, A mathematical theory of communication, *Bell System Technical Journal* 27 (4) (1948) 623–656 (1948).
- [101] M. Rosvall, D. Axelsson, T. C. Bergstrom, The map equation, *Eur. Phys. J. Special Topics* 178 (1) (2009) 13–23 (2009). doi:10.1140/Epjst/E2010-01179-1.
- [102] T. Kawamoto, M. Rosvall, Estimating the resolution limit of the map equation in community detection, *Phys. Rev. E* 91 (2015) 012809 (Jan 2015). doi:10.1103/Physreve.91.012809.
- [103] M. Rosvall, C. T. Bergstrom, Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems, *PLOS ONE* 6 (4) (2011) 1–10 (04 2011). doi:10.1371/journal.pone.0018209.
- [104] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, *Knowl. Inform. Syst.* 42 (1) (2015) 181–213 (Jan 2015). doi:10.1007/S10115-013-0693-Z.
- [105] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, *Phys. Rev. E* 78 (2008) 046110 (Oct 2008). doi:10.1103/Physreve.78.046110.
- [106] S. Maslov, K. Sneppen, Specificity and stability in topology of protein networks, *Science* 296 (5569) (2002) 910–913 (2002).
- [107] H. Jeong, S. P. Mason, A.-L. Barabási, Z. N. Oltvai, Lethality and centrality in protein networks, *Nature* 411 (6833) (2001) 41–42 (2001).
- [108] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, et al., A comprehensive analysis of protein–protein interactions in *saccharomyces cerevisiae*, *Nature* 403 (6770) (2000) 623–627 (2000).
- [109] G. Amitai, A. Shemesh, E. Sitbon, M. Shklar, D. Netanel, I. Venger, S. Pietrokovski, Network analysis of protein structures identifies functional residues, *Journal of molecular biology* 344 (4) (2004) 1135–1146 (2004).
- [110] W. Yan, J. Zhou, M. Sun, J. Chen, G. Hu, B. Shen, The construction of an amino acid network for understanding protein structure and function, *Amino acids* 46 (6) (2014) 1419 (2014).
- [111] P. Csermely, T. Korcsmáros, H. J. Kiss, G. London, R. Nussinov, Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review, *Pharmacology & therapeutics* 138 (3) (2013) 333–408 (2013).
- [112] B. Chakrabarty, N. Parekh, Naps: Network analysis of protein structures, *Nucleic acids research*

- 44 (W1) (2016) W375–W382 (2016).
- [113] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, T. Ideker, Cytoscape: a software environment for integrated models of biomolecular interaction networks, *Genome research* 13 (11) (2003) 2498–2504 (2003).
- [114] N. T. Doncheva, K. Klein, F. S. Domingues, M. Albrecht, Analyzing and visualizing residue networks of protein structures, *Trends in biochemical sciences* 36 (4) (2011) 179–182 (2011).
- [115] J. C. Delvenne, S. N. Yaliraki, M. Barahona, Stability of graph communities across time scales, *P. Natl. Acad. Sci. USA* 107 (29) (2010) 12755–12760 (Jul 20 2010). doi:10.1073/Pnas.0903215107.
- [116] A. Delmotte, E. W. Tate, S. N. Yaliraki, M. Barahona, Protein multi-scale organization through graph partitioning and robustness analysis: application to the myosin–myosin light chain interaction, *Physical biology* 8 (5) (2011) 055010 (2011).
- [117] S. Tasdighian, L. Di Paola, M. De Ruvo, P. Paci, D. Santoni, P. Palumbo, G. Mei, A. Di Venere, A. Giuliani, Modules identification in protein structures: the topological and geometrical solutions, *Journal of chemical information and modeling* 54 (1) (2013) 159–168 (2013).
- [118] H. J. Feldman, Identifying structural domains of proteins using clustering, *BMC bioinformatics* 13 (1) (2012) 286 (2012).
- [119] J. S. Hleap, E. Susko, C. Blouin, Defining structural and evolutionary modules in proteins: a community detection approach to explore sub-domain architecture, *BMC structural biology* 13 (1) (2013) 20 (2013).
- [120] M. Szalay-Bekő, R. Palotai, B. Szappanos, I. A. Kovács, B. Papp, P. Csermely, Moduland plug-in for cytoscape: determination of hierarchical layers of overlapping network modules and community centrality, *Bioinformatics* 28 (16) (2012) 2202–2204 (2012).
- [121] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, Jupyter notebooks – a publishing format for reproducible computational workflows, in: F. Loizides, B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, pp. 87 – 90 (2016).
- [122] Y. Li, A. Roy, Y. Zhang, Haad: a quick algorithm for accurate prediction of hydrogen atoms in protein structures, *PloS one* 4 (8) (2009) e6701 (2009).
- [123] M. Heinig, D. Frishman, Stride: a web server for secondary structure assignment from known atomic coordinates of proteins, *Nucleic acids research* 32 (suppl\_2) (2004) W500–W502 (2004).
- [124] B. Cordero, V. Gomez, A. E. Platero-Prats, M. Reves, J. Echeverria, E. Cremades, F. Barragan, S. Alvarez, Covalent radii revisited, *Dalton Trans.* (2008) 2832–2838 (2008). doi:10.1039/B801115J.
- [125] Y. Hu, Efficient, high-quality force-directed graph drawing, *Mathematica Journal* 10.1 (2005) 37–71 (2005).
- [126] T. P. Peixoto, The graph-tool python library, *Figshare* (2014). doi:10.6084/M9.Figshare.1164194.

- [127] A. A. Zavitsas, The relation between bond lengths and dissociation energies of carbon-carbon bonds, *J. Phys. Chem. A* 107 (6) (2003) 897–898 (2003). doi:10.1021/Jp0269367.
- [128] L. Schrödinger, The Pymol molecular graphics system, version 1.8 (2015).
- [129] A. Arenas, A. Fernandez, S. Gomez, Analysis of the structure of complex networks at different resolution levels, *New J. Phys.* 10 (May 29 2008). doi:10.1088/1367-2630/10/5/053039.
- [130] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, M. Barahona, Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit, *PloS one* 7 (2) (2012) e32210 (2012).
- [131] W. P. Grant, S. E. Ahnert, Modular decomposition of protein structure using community detection, *Journal of Complex Networks* 7 (1) (2019) 101–113 (02 2019). doi:10.1093/comnet/cny014.
- [132] S. Mir, Y. Alhroub, S. Anyango, D. R. Armstrong, J. M. Berrisford, A. R. Clark, M. J. Conroy, J. M. Dana, M. Deshpande, D. Gupta, A. Gutmanas, P. Haslam, L. Mak, A. Mukhopadhyay, N. Nadzirin, T. Paysan-Lafosse, D. Sehnal, S. Sen, O. S. Smart, M. Varadi, G. J. Kleywegt, S. Velankar, PDBe: towards reusable data delivery infrastructure at protein data bank in Europe, *Nucleic Acids Research* 46 (D1) (2017) D486–D492 (11 2017). doi:10.1093/nar/gkx1070.
- [133] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *Journal of computational and applied mathematics* 20 (1987) 53–65 (1987).
- [134] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2) (2001) 411–423 (2001).
- [135] J. Chandonia, G. Hon, N. S. Walker, L. Lo Conte, P. Koehl, M. Levitt, S. E. Brenner, The astral compendium in 2004, *Nucleic Acids Research* 32 (suppl1) (2004) D189–D192 (01 2004). doi:10.1093/nar/gkh034.
- [136] J.-M. Chandonia, N. K. Fox, S. E. Brenner, SCOPe classification of large macromolecular structures in the structural classification of proteins-extended database, *Nucleic Acids Research* 47 (D1) (2018) D475–D481 (11 2018). doi:10.1093/nar/gky1134.
- [137] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* 12 (2011) 2825–2830 (2011).
- [138] S. E. Ahnert, W. P. Grant, C. J. Pickard, Revealing and exploiting hierarchical material structure through complex atomic networks, *NPJ Comp. Mat.* (2017).
- [139] N. M. O’Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, G. R. Hutchison, Open babel: An open chemical toolbox, *J. Cheminformatics* 3 (Oct 7 2011). doi:10.1186/1758-2946-3-33.
- [140] A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using networkx, in: *Scipy2008*, 2008, pp. 11–15 (Aug 2008).
- [141] M. Bostock, V. Ogievetsky, J. Heer, D<sup>3</sup> data-driven documents, *IEEE transactions on visualization and computer graphics* 17 (12) (2011) 2301–2309 (2011).

- 
- [142] H. Bunke, K. Shearer, A graph distance metric based on the maximal common subgraph, *Pattern recognition letters* 19 (3) (1998) 255–259 (1998).
- [143] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, An improved algorithm for matching large graphs, in: *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, 2001, pp. 149–159 (2001).
- [144] G. James, D. Witten, T. Hastie, R. Tibshirani, *An introduction to statistical learning*, Vol. 112, Springer, 2013 (2013).
- [145] A. Bairoch, The enzyme database in 2000, *Nucleic acids research* 28 (1) (2000) 304–305 (2000).
- [146] L. Pu, R. G. Govindaraj, J. M. Lemoine, H.-C. Wu, M. Brylinski, Deepdrug3d: Classification of ligand-binding pockets in proteins with a convolutional neural network, *PLoS computational biology* 15 (2) (2019) e1006718 (2019).
- [147] T. Fawcett, An introduction to ROC analysis, *Pattern recognition letters* 27 (8) (2006) 861–874 (2006).
- [148] D. E. Knuth, *Fundamental Algorithms*, 3rd Edition, Addison-Wesley, 1997, pp. 66–70 (1997).
- [149] C. P. Massen, J. P. K. Doye, Identifying communities within energy landscapes, *Phys. Rev. E* 71 (4, 2) (Apr 2005). doi:10.1103/Physreve.71.046101.
- [150] A. Buckley, The hepthesis  $\text{\LaTeX}$  class.

# List of Figures

2.1	Schematic representation of amino acids, showing peptide bond formation	5
2.2	Schematic of the SOAP descriptor generation process . . . . .	11
2.3	An example graph with 15 nodes and 23 edges . . . . .	13
2.4	Degree distributions . . . . .	15
2.5	Average Path Length . . . . .	16
2.6	Average Clustering . . . . .	17
2.7	An example of a network with community structure . . . . .	18
2.8	Examples of cliques, strong communities, and weak communities . . . . .	19
3.1	Logic diagram for the codebase . . . . .	30
3.2	An example of the structure of a PDB File . . . . .	31
3.3	Logic diagram for the network generation process . . . . .	32
3.4	An example of the edges generated using STRIDE . . . . .	34
3.5	Comparison between HAAD output and original PDB file. . . . .	34
3.6	Examples of high and low scaling . . . . .	35
3.7	The test proteins used . . . . .	37
3.8	The distributions of closeness centrality for 6 test proteins . . . . .	39
3.9	The distributions of clustering coefficient for 6 test proteins . . . . .	40
3.10	Variation of network size with cutoff radius . . . . .	41
3.11	Run time of Rust code vs Python code . . . . .	42
3.12	An example of visualisation methods . . . . .	43
3.13	Comparison between results of SBM and Infomap on the same networks .	44

---

3.14	1PKN . . . . .	45
3.15	The variation of conductance with Infomap repetitions . . . . .	46
4.1	Modified Jaccard Test . . . . .	49
4.2	Randomly generated null models . . . . .	50
4.3	Comparison between Pfam domains and Infomap results for three test proteins . . . . .	51
4.4	z-scores showing the significance of the match between Pfam and Infomap . . . . .	53
4.5	The relationship between the modified Jaccard index and the conductance of the Pfam domain . . . . .	53
4.6	Alignment of SCOP domain for three test structures . . . . .	54
4.7	The SCOP domain compared to the overall structure. . . . .	55
4.8	The silhouette score against dendrogram level for the c.1.8 test set. . . . .	56
4.9	All-to-all similarity score for a test set of SCOP domains . . . . .	57
4.10	Violin plot of module sizes for a test set of SCOP domains . . . . .	58
4.11	Violin plot showing intra-cluster fragment RSMD vs whole-chain RSMD for a test set of SCOP domains . . . . .	58
4.12	Alignment of three generated fragments . . . . .	59
4.13	Comparison between SCOP domain and Infomap module . . . . .	60
4.14	Comparison of test-set community structure to existing annotations . . . . .	60
4.15	Hierarchical clustering of similarity scores for a large multi-domain test set . . . . .	62
4.16	Silhouette score against dendrogram threshold. . . . .	62
4.17	Dendrogram at optimal threshold. . . . .	63
4.18	Histogram of fragment sizes for each cluster. . . . .	64
4.19	Selected protein fragments from the multi-domain test set, in Cluster 0. . . . .	65
4.20	Cluster 1. . . . .	65
4.21	Cluster 2. . . . .	66
4.22	Overlap between the generated and expected annotations. . . . .	66
4.23	1BF2 . . . . .	68

4.24	A z-score comparison for modularity+correlation and Infomap . . . . .	69
4.25	Compression vs average module size for single proteins . . . . .	71
4.26	Compression vs average module size for sets of proteins . . . . .	72
5.1	Example of a supernetwork . . . . .	74
5.2	Protein classes resulting from multi-chain decomposition . . . . .	75
5.3	Super-networks from a non-redundant subset of the PDB. . . . .	76
5.4	Comparison between two SCOP folds, showing the number of communities	77
5.5	Network similarity vs structure similarity . . . . .	79
5.6	The TM-score between proteins with isomorphic super-networks, versus the TM-score between the full dataset . . . . .	79
5.7	Example of multi level community structure . . . . .	81
5.8	Example of multi level community structure . . . . .	81
5.9	Example of a tree . . . . .	81
5.10	Example of the tree labelling process . . . . .	82
5.11	Gallery of the labelled networks. . . . .	84
6.1	The clustermap for SCOP classes A and B. . . . .	87
6.2	Two test proteins from SCOP classes <b>a</b> and <b>b</b> . . . . .	88
6.3	The clustermap for SCOP classes C and D. . . . .	89
6.4	The clustermap for the five EC classes. . . . .	90
6.5	Schematic of the structure-identifying process. . . . .	92
6.6	The similarity-highlighting algorithm on 2d2m. . . . .	93
6.7	The similarity-highlighting algorithm on 1x9f. . . . .	93
6.8	The similarity-highlighting algorithm on 1X9f, showing the sidechain problem. . . . .	94
6.9	The similarity-highlighting algorithm on 1X9f, with 20-neighbour smoothing.	95
6.10	A schematic of the ROC curve. . . . .	96
6.11	ROC curves for selected proteins in the test set. . . . .	97



---

6.12	The average ROC curve for the full test set. . . . .	98
A.1	A decomposition of 1UBQ into modules using the AFG method . . . . .	103
A.2	The distributions of betweenness centrality for 6 test proteins . . . . .	105
A.3	The distributions of degree centrality for 6 test proteins . . . . .	105
A.4	The distributions of eigenvector centrality for 6 test proteins . . . . .	105
A.5	The distributions of Katz centrality for 6 test proteins . . . . .	106
A.6	The distributions of pagerank scores for 6 test proteins . . . . .	106
A.7	Selected protein fragments from the multi-domain test set, in Cluster 0. .	107
A.8	Cluster 1. . . . .	107
A.9	Cluster 2. . . . .	108
A.10	Cluster 3. . . . .	108
A.11	Cluster 4. . . . .	109
A.12	Cluster 5. . . . .	109
A.13	Cluster 6. . . . .	110
A.14	Cluster 7. . . . .	110
A.15	Cluster 8. . . . .	110
A.16	Cluster 9. . . . .	111

# Colophon

This thesis was made in  $\text{\LaTeX}$  using a modified version of the “hepthesis” class [150]. The  $\text{\TeX}$  was edited and compiled using the LaTeX Workshop extension to Visual Studio Code.